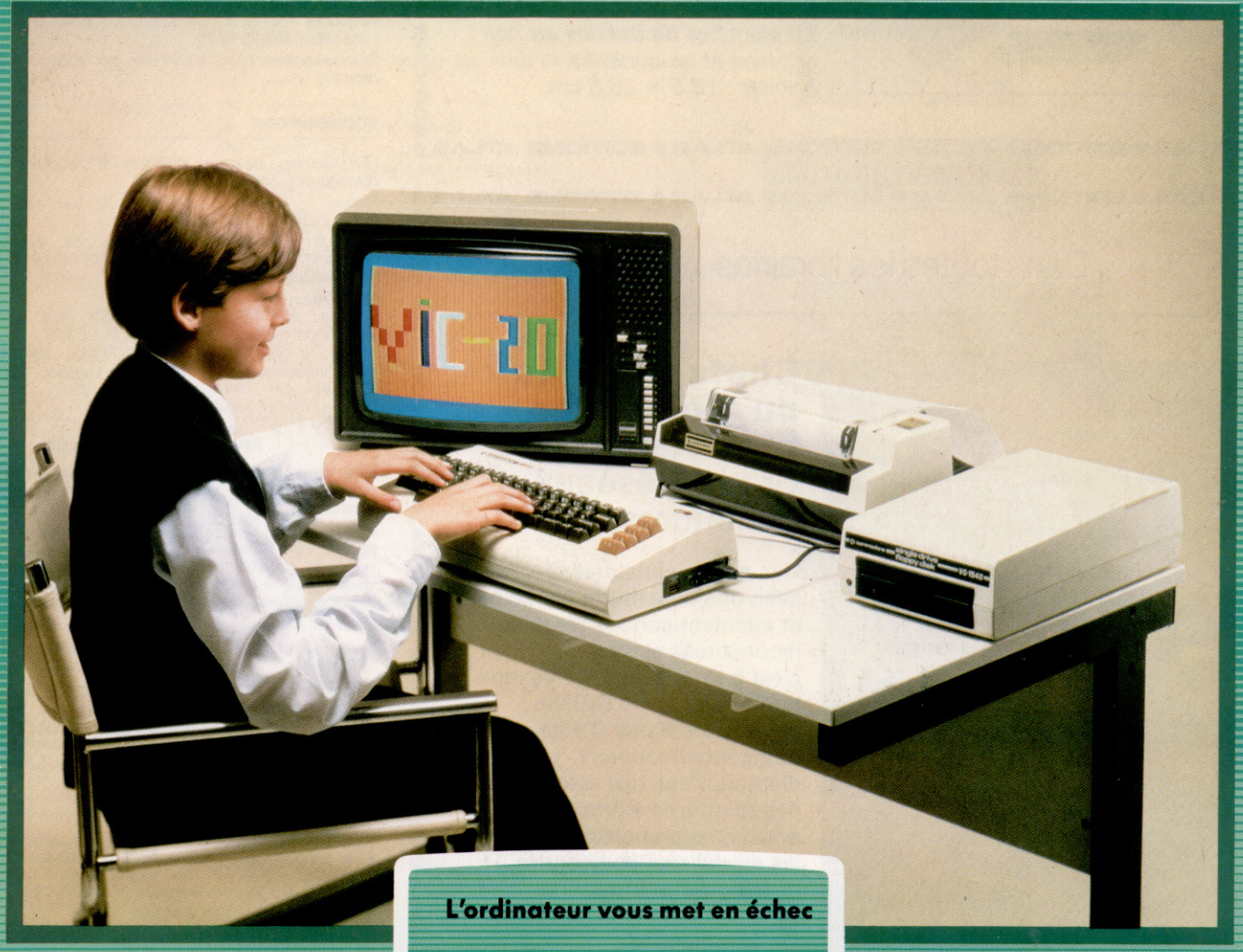
ABC INFORMATIQUE

N° 40

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



L'ordinateur vous met en échec

Le Colour Genie

Il faut creuser l'idée !

Silence profond

EDITIONS ATLAS

M6062-40-12F 85FB-3,80FS-$1.95

Dans toutes les librairies

La maison de Lizzie

Livre pratique, original et particulièrement attrayant que celui de Lizzie Napoli. Il décrit la maison idéale, qui est d'ailleurs un peu la sienne. Avec force détails, dessins et photographies, Lizzie évoque ce qu'est pour elle la maison heureuse et suggère des adaptations simples.
Les aménagements proposés par *La Maison de Lizzie* apporteront une touche de fantaisie dans votre intérieur. Et le charme de la poésie.

Un volume relié,
sous couverture cartonnée.
96 pages.
24 photos en couleurs.
70 planches de dessins en noir et blanc et en couleurs.
Format : 18,5 × 26,8 cm.

Dans toutes les librairies

Chefs-d'œuvre du modélisme ferroviaire

Grands réseaux miniatures

Des trains miniatures réalisés avec une exactitude hallucinante et circulant sur des réseaux reconstitués scrupuleusement : c'est à un merveilleux voyage que nous convie ce livre.
De nos tortillards d'antan aux trains internationaux d'aujourd'hui, ces réseaux, qui dégagent une extraordinaire poésie, surprendront aussi bien les modélistes chevronnés que les débutants.

Un volume relié.
Jaquette illustrée. 144 pages.
169 photos en couleurs.
20 schémas.
Format : 22 x 28,5 cm.

aBc INFORMATIQUE

cours d'informatique pratique et familiale

EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 28-10-10. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS, Z.I. de Lucé, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume IV, n° 40.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Remy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Claire Bischoff *(correction)*.
Crédit photographique, couverture : Procef.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : octobre 1984. 198410. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Échecs sur ordinateur

Les jeux d'échecs ont retenu l'attention de la plupart des programmeurs. Nous examinons ici les principes de conception des programmes de jeu d'échecs sur ordinateur.

Champion d'échecs
David Levy est un champion international qui a abandonné la compétition officielle en 1978. Il avait fait le pari en 1968 qu'aucun programme d'échecs sur ordinateur ne pourrait le battre au cours des dix prochaines années. Depuis lors, la période du pari a été dépassée et il demeure toujours invaincu. Il est aujourd'hui l'un des plus grands spécialistes en ce qui concerne les jeux d'échecs sur ordinateur, et il dirige Intelligent Software, une société responsable des techniques de programmation que l'on retrouve dans de nombreux programmes d'échecs sur ordinateur et sur micro-ordinateur. Levy croit que les micros s'approchent de plus en plus des performances des gros ordinateurs. Il estime qu'avant huit ans un micro sera capable de battre Belle (une machine dédiée au jeu d'échecs) et le gros ordinateur Cray Blitz (qui remporta le championnat mondial d'échecs sur ordinateur en 1983) en utilisant probablement des microprocesseurs parallèles pour accélérer les fonctions de recherche.

Peu de jeux ont exercé autant d'attrait que les échecs : ce jeu est pratiqué par des millions de personnes dans le monde depuis des centaines d'années et ses règles demeurent inchangées depuis le XVII[e] siècle. Nombreux sont ceux qui ont consacré leur vie à l'étude et à la maîtrise de ce jeu de stratégie, trouvant satisfaction dans la rigueur et la finesse requises. Diverses variantes du jeu d'échecs sont apparues, introduisant un niveau supérieur de complexité : par exemple, le jeu d'échecs tridimensionnel utilise plusieurs tableaux suspendus et demande beaucoup plus de concentration. Le jeu d'échecs à trois est une autre variante qui est jouée sur un tableau ayant la forme d'un Y. Sur les diagonales où se situe l'intersection des trois « ailes », des règles spéciales régissent le mouvement des pièces (deux joueurs s'associent contre le troisième et tentent d'arracher la victoire). Mais aucune de ces versions n'a réussi à surpasser en qualité la confrontation fascinante que permet le jeu à deux personnes sur un tableau de 64 carrés.

La raison en est l'incroyable complexité du jeu lui-même. En 1949, le mathématicien Claude Shannon écrivit un livre intitulé « Programmation d'un ordinateur pouvant jouer aux échecs », dans lequel il calcula qu'il existait plus de 10^{120} jeux différents de 40 mouvements. Cela signifie qu'une personne jouant aux échecs vingt-quatre heures par jour, et consacrant une heure à chaque jeu (ce qui n'est pas long pour 40 mouvements) devrait jouer pendant plus de 10^{17} années pour passer en revue tous les jeux possibles. Évidemment, la théorie des échecs a été si bien analysée jusqu'à maintenant que ce nombre de possibilités a, en pratique, diminué selon un facteur qui dépend de l'expérience du joueur.

Il n'est pas étonnant que les programmes permettant aux ordinateurs de jouer aux échecs soient très longs et très difficiles à écrire. Des programmes d'échecs sont exécutés sur de gros ordinateurs depuis longtemps et il existe maintenant de nombreuses versions destinées aux micro-ordinateurs domestiques. Le développement de programmes d'échecs de bonne qualité dépend des innovations techniques du matériel et de la qualité des programmes.

Stratégies programmées

Afin d'évaluer certains des programmes d'échecs les plus populaires pour micros domestiques, « ABC Informatique » a organisé un mini-tournoi pour ces produits : Sargon III, tournant sur un Apple IIe (disquette vendue 600 F — Hayden Software, programmé par Dan et Kathe Spracklen); Cyrus IS Chess sur un Spectrum de 48 K (cassette vendue 120 F — Sinclair Software, programmé par Intelligent Software); Colossus 2.0 sur un Commodore 64 (disquette vendue 150 F — CDS MicroSystems, programmé par Martin Bryan); et Grand Master 64, aussi pour le Commodore 64 (cassette vendue 80 F — Audiogenic, programmé par Kingsoft).

Bien que ces programmes se soient déjà mesurés dans un tournoi international d'échecs sur micros, nous désirions effectuer notre propre évaluation basée sur la facilité d'utilisation et sur l'efficacité.

L'un des problèmes posés par ce genre de compétition est la difficulté de déterminer quels sont les niveaux de jeux qui placent deux programmes sur un pied d'égalité. Ils sont généralement définis en fonction de la période de réflexion accordée à chaque programme lors d'un mouvement, mais il peut ne pas exister de corrélation directe entre une période de 10 secondes dans un programme et le même laps de temps dans un autre. Par exemple, Sargon III continue à analyser le jeu pendant le mouvement de l'adversaire. Tous les autres programmes mettent leurs générateurs de déplacement hors fonction à cet instant. Nous nous sommes néanmoins efforcés de choisir judicieusement les paires d'opposants.

Qualité du jeu

Généralement, tous les programmes eurent un jeu plutôt décevant au niveau inférieur (réflexion d'environ 10 secondes par mouvement). Et tous firent des mouvements très curieux, apparemment inutiles au milieu du jeu. Cela était probablement dû à une position « tranquille », où les ordinateurs ont simplement perdu leur temps en attendant que quelque chose d'intéressant se produise. A un niveau de compétition plus élevé (environ 10 minutes par mouvement), les quatre programmes donnèrent un jeu tactique plus intelligent et parfois brillant. Les résultats du tournoi sont illustrés dans le tableau ci-contre.

Puisque les ordinateurs sont essentiellement des calculateurs rapides, le jeu d'échecs sur ordinateur se déroule à partir de calculs numériques qui servent à évaluer les deux éléments essentiels du jeu : le matériel et la mobilité. Le « matériel » d'un jeu est le nombre et la force des pièces sur le jeu. Le programme d'échecs attribue une valeur numérique à chaque pièce. Le roi peut recevoir une valeur infinie ou une valeur arbitrairement élevée, comme 10 000 (puisque la perte du roi implique la fin du jeu); une valeur de neuf est attribuée à la reine; la tour vaut cinq; les fous et les cavaliers trois; et les pions un. Lorsqu'il doit décider s'il est rentable de perdre une pièce pour s'emparer de l'une des pièces de son adversaire, l'ordinateur compare leurs valeurs respectives. La plupart des programmes d'échecs sur ordinateur accordent une grande importance aux valeurs relatives et consentent rarement à faire un mouvement qui s'avère désavantageux sur le plan « matériel », sauf si ce mouvement améliore ses positions.

La mobilité est très importante aux échecs puisque toute pièce n'a que peu de poids stratégique si son mouvement est restreint. Inversement, sa valeur croît si sa position lui permet d'exercer une influence à plusieurs endroits à la fois. Le programme d'échecs doit donc être en mesure d'évaluer à la fois la mobilité et la valeur intrinsèque de chaque pièce. De plus, le programme doit être capable de planifier une stratégie, de déterminer quelle sera la meilleure séquence de mouvements à partir d'une position donnée. C'est ici que les programmes de jeu d'échecs peuvent exceller, en utilisant la vitesse de l'ordinateur pour examiner un grand nombre de possibilités de mouvements dans un temps très court.

La plupart des programmes d'échecs utilisent une technique dite de « force brute », qui permet d'analyser le maximum de mouvements dans le temps alloué. La période allouée est fonction du « niveau » sélectionné au début du jeu; chaque niveau accorde un temps de réflexion différent à l'ordinateur. Ces périodes peuvent aller de quelques secondes à plusieurs heures, et plus le temps de recherche autorisé est long, meilleures sont les chances pour l'ordinateur de découvrir la bonne stratégie d'attaque.

Lors de chaque mouvement, l'ordinateur détermine si le roi est ou n'est pas en échec, et examine la possibilité de s'emparer ou de perdre des pièces, d'occuper des positions stratégiques, et étudie plusieurs autres questions similaires. La qualité du résultat est directement proportionnelle au nombre de critères examinés. La question ultime est de découvrir comment le roi de l'adversaire peut être mis en échec. Dans les jeux opposant les ordinateurs à des êtres humains, les ordinateurs ont un avantage marqué au niveau de la vitesse et du domaine de recherche — un excellent joueur humain devrait

Cyrus IS Chess égalisa avec Colossus, battit Grand Master au niveau inférieur et fit match nul avec Sargon III au niveau supérieur.

Colossus égalisa avec Cyrus IS Chess au niveau inférieur, battit Grand Master au niveau supérieur, et égalisa avec Sargon III au niveau supérieur.

Sargon III perdit contre Grand Master au niveau inférieur et fit match nul contre Cyrus et contre Colossus au niveau supérieur.

Grand Master battit Sargon III et perdit contre Cyrus IS Chess au niveau inférieur, et perdit contre Colossus au niveau supérieur.

Fonctions

Tous les bons programmes d'échecs peuvent roquer, échanger un pion contre une reine et comprendre une situation d'impasse. Certains de ces programmes comportent des fonctions additionnelles intéressantes. Sargon III est le programme qui offre le plus de fonctions complémentaires et comprend une seconde disquette qui renferme 107 matchs d'échecs classiques et 45 problèmes d'échecs parmi les plus intéressants. La documentation est superbe et compte 75 feuilles détachables. Sargon III est exécuté sur un Apple IIe et coûte trois fois plus cher que les autres programmes.

Fonction	Grand Master 64	Colossus	Cyrus IS	Sargon III
Jeu invisible	NON	OUI	NON	NON
Mouvements possibles	NON	OUI	NON	NON
Meilleur prochain mouvement	NON	NON	OUI	NON
Répète un jeu	NON	OUI	OUI	NON
Affiche le listage	NON	OUI	NON	OUI
Imprime le listage	NON	NON	OUI	OUI
Revient à des mouvements précédents	OUI	OUI	OUI	OUI
Travaux pratiques	OUI	NON	NON	OUI
Transformation de pièces	NON	OUI	OUI	OUI
Jeu humain	NON	OUI	OUI	OUI
Changement de côté	OUI	OUI	OUI	OUI
Analyse de problèmes	NON	OUI	OUI	OUI
Illustre recherche	Un essai	OUI	NON	OUI
Inversion du tableau	OUI	OUI	OUI	OUI
Tirage	NON	NON	NON	OUI
Impression du tableau	NON	NON	OUI	OUI
Sauvegarde du jeu	NON	OUI	OUI	OUI
Interdit la bibliothèque	NON	NON	NON	OUI
Jeu automatique	NON	OUI	OUI	OUI
Horloge temps réel	OUI	OUI	NON	OUI

Conclusion

Cyrus IS et Colossus sont les jeux les plus faciles à utiliser parce que les mouvements sont entrés à l'aide du curseur, tandis qu'avec Sargon III et Grand Master vous devez entrer les mouvements en spécifiant des coordonnées, comme E2-E4. Colossus et Sargon ont les meilleurs affichages.

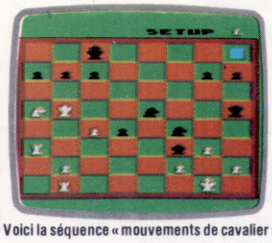
toujours être en mesure de battre un excellent programme d'échecs en raison de la possibilité qu'a l'être humain de créer de nouvelles ouvertures et de nouvelles positions. Les ordinateurs pratiquent un excellent jeu tactique; mais même parmi les champions d'échecs, un joueur qui excelle au niveau du positionnement des pièces devrait toujours battre un bon joueur tactique. Les programmeurs de jeux d'échecs sur ordinateur se sont surtout concentrés sur le plan tactique puisque, pour un ordinateur, un jeu tactique n'implique que de simples calculs. Il arrive souvent que l'ordinateur ne sache quelle suite donner à des mouvements non conventionnels menés par un adversaire humain.

Un style de programmation récemment mis au point implique une « recherche sélective ». A l'aide de cette technique, l'ordinateur peut imiter l'être humain en examinant à fond un petit nombre de mouvements possibles. Les Allemands Hegener et Glaser ont utilisé la technique de recherche sélective dans leur programme Mephisto III, qui examine tous les mouvements offerts lors des deux premiers mouvements, puis limite la recherche et n'examine à fond qu'un petit nombre de mouvements. Mephisto III tente aussi d'établir une distinction entre des positions tranquilles et tactiques. Avec de telles techniques, les ordinateurs devraient éventuellement proposer un véritable défi aux joueurs humains.

Le cœur contre la raison

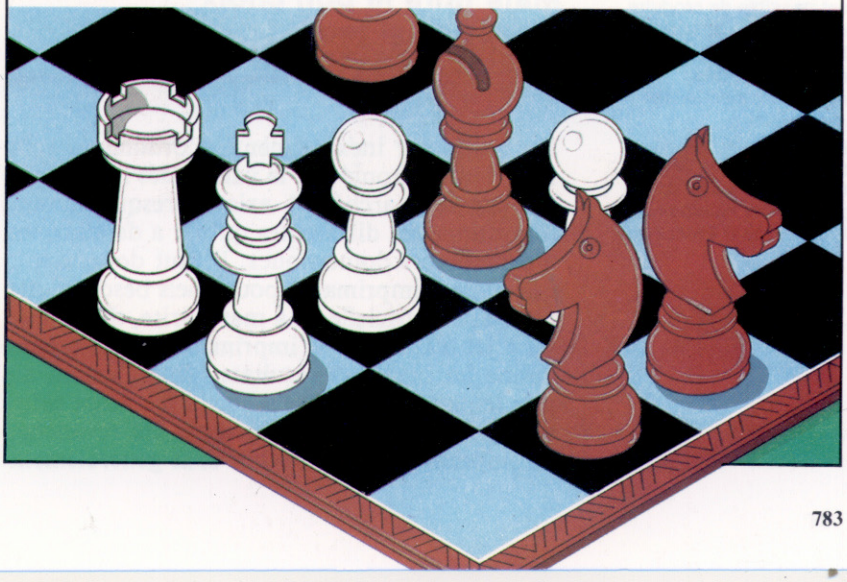
La possibilité d'analyser d'avance chaque position jusqu'à neuf mouvements donne un avantage certain aux ordinateurs par rapport aux humains.

Ici, Moritz (noir) joue contre Emmerich (blanc) en 1922; ce jeu est utilisé dans le film « Night Moves ». Le noir peut mettre en échec en sacrifiant sa reine, et en faisant ensuite trois excellents mouvements de cavalier; la plupart des joueurs humains auraient sans hésitation choisi cette séquence. Moritz ne la vit pas et le regretta sans doute amèrement. Tous nos programmes réussirent l'échec et mat, mais aucun n'utilisa la stratégie des mouvements de cavalier, bien que certains aient pu l'envisager. L'incapacité dont souffrent les ordinateurs à percevoir cette solution comme la meilleure semble offrir aux joueurs humains le seul moyen de défense.



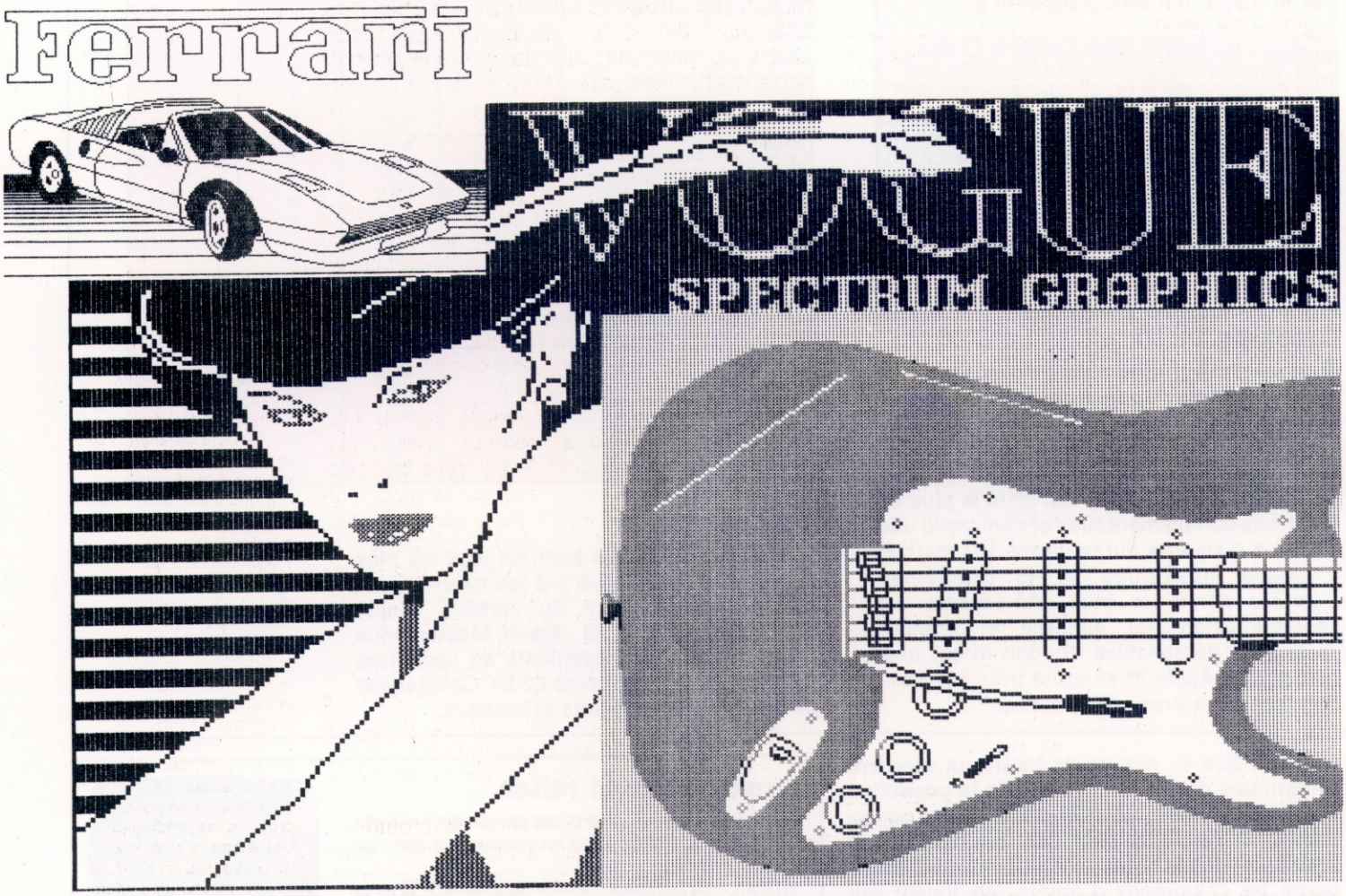
Voici la séquence « mouvements de cavalier »

1		H5–H2 ch
2	G1–H2	E5–G4 ch
3	H2–G1	F4–H3 ch
4	G1–F1	G4–H2 mat



Roy Ingram

Imprimante et matrice



Imprimante artistique
Cet exemple illustre le type de graphiques que peuvent produire certaines imprimantes matricielles. Chaque aiguille de la tête d'impression est commandée individuellement, et il est possible de produire des dessins complexes. Nous donnerons ultérieurement les explications détaillées concernant la production de tels graphiques. Ces images furent créées à l'aide du programme Paintbox de Print'n'Plotter Products.

La plupart des utilisateurs d'ordinateur domestique décident éventuellement d'acheter une imprimante pour compléter leur système et simplifier le travail. Mais faire le bon choix n'est pas si facile!

Un utilisateur inexpérimenté d'ordinateur sera sans doute dérouté par la variété des imprimantes sur le marché. Il existe presque autant d'imprimantes différentes qu'il y a de marques d'ordinateurs domestiques. Il faut donc savoir quel type d'imprimante pour quels besoins; elle peut être matricielle ou à marguerite, thermique ou à jet d'encre. Une imprimante à marguerite donne les meilleurs résultats (généralement au prix le plus élevé) et constitue la meilleure solution pour le traitement de texte. En revanche, les imprimantes matricielles sont généralement meilleur marché, plus rapides et idéales pour les listages et pour des tâches de programmation générales.

Une imprimante matricielle peut coûter moins de 2 500 F, bien que des modèles plus sophistiqués dépassent les 12 000 F. Les critères importants dont il faut tenir compte sont la vitesse d'impression et la qualité du texte imprimé; les modèles plus coûteux offrent des fonctions additionnelles comme l'espacement proportionnel (c'est-à-dire que des caractères étroits comme les « i » occupent moins d'espace que des caractères larges commes les « m ») et un choix de polices de caractères.

La vitesse d'impression est importante puisque l'utilisation de l'imprimante mobilise l'ordinateur : le texte doit être stocké dans la mémoire jusqu'à ce que l'imprimante soit prête à l'imprimer. Par conséquent, l'ordinateur ne peut être utilisé à d'autres tâches quand une impression est en cours. Les vitesses d'impression sont exprimées en « caractères par seconde » (cps). Ainsi, un modèle coûteux fonctionnant à 200 cps peut prendre une minute pour imprimer un long listage de programme, tandis qu'une

imprimante moins coûteuse avec une vitesse de 30 cps pourrait prendre plus de six minutes pour produire le même listage. Ce problème peut être résolu en utilisant un tampon d'imprimante. Il s'agit simplement d'une carte renfermant des puces RAM qui est connectée entre l'imprimante et l'ordinateur et qui stocke les données pendant le travail de l'imprimante, ce qui permet à l'ordinateur de se consacrer à d'autres opérations. Les imprimantes plus coûteuses possèdent des tampons intégrés importants.

La qualité d'impression varie considérablement d'une imprimante à l'autre. Elle dépend principalement du nombre d'aiguilles que compte la tête d'impression (le mécanisme qui forme les caractères sur le papier). Les modèles les moins coûteux n'utilisent que 7 aiguilles, tandis que d'autres modèles plus coûteux en ont 16 ou plus. Sur l'imprimante Commodore, qui n'a que sept aiguilles, les caractères sont produits sous la forme d'une matrice de sept points par sept. La Canon PW1080 utilise cependant une matrice de 16 × 23 pour produire ses caractères. Résultats : les points individuels sont invisibles et les caractères ont une excellente définition. Pour des listages de programme, la qualité d'impression n'a pas de réelle importance, mais elle est essentielle en traitement de texte.

Une imprimante matricielle est en fait un micro-ordinateur dédié; elle utilise des puces ROM et RAM et possède un microprocesseur. Elle peut donc être programmée pour faire autre chose que simplement imprimer du texte. C'est possible en envoyant des codes de commande spéciaux ou en réglant de petits interrupteurs — nommés interrupteurs DIP (Dual In-line Package, double rangée de connexions) — situés à l'intérieur de l'imprimante. Par exemple, le jeu de caractères ASCII standard qui est stocké dans la mémoire peut être modifié pour s'adapter à différents alphabets. En Grande-Bretagne, le signe dièse (#) est souvent remplacé par un signe de livre anglaise (£).

Parmi les autres effets pouvant être produits, citons l'impression en caractères gras, l'impression de caractères double largeur et l'emploi de différents espacements d'interligne. L'Epson FX80 est l'une des imprimantes matricielles les plus souples; elle possède plus de soixante-dix fonctions d'impression. Elle peut imprimer des caractères italiques, souligner le texte automatiquement et autorise un espacement proportionnel.

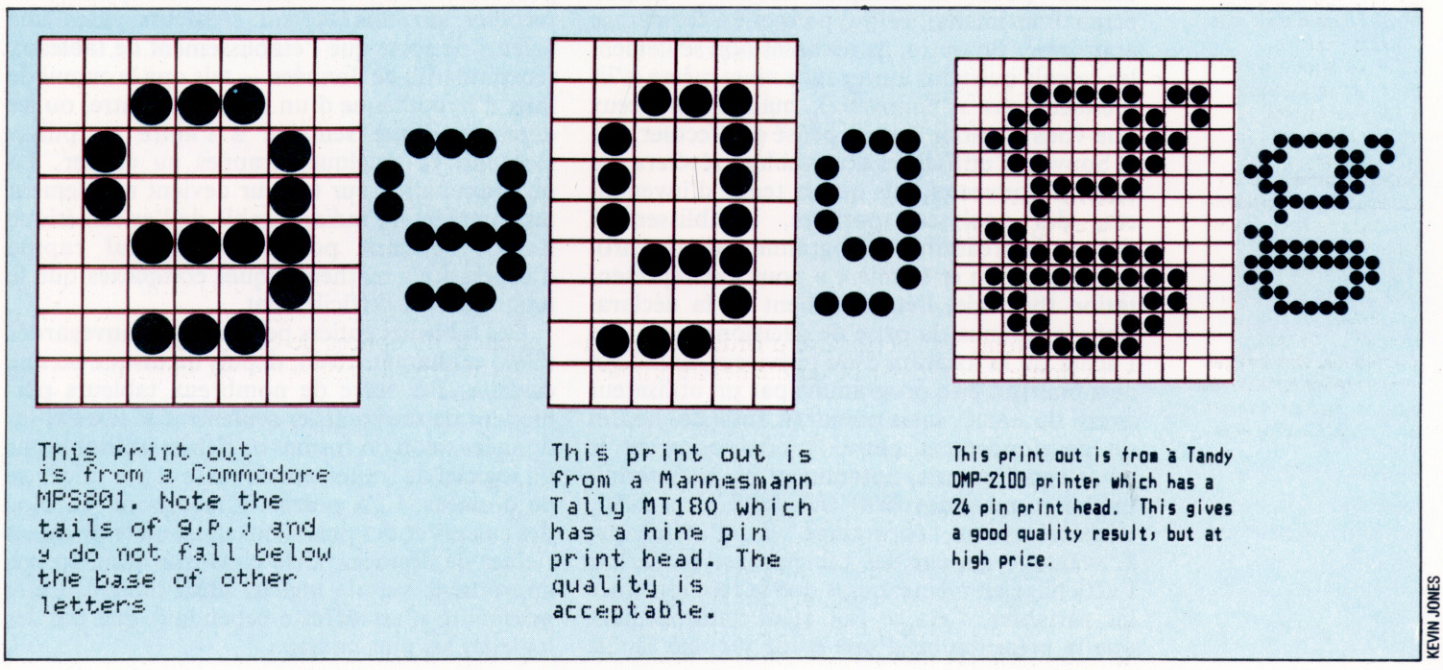
La gamme d'imprimantes Epson constitue une véritable norme dans l'industrie informatique. Cela signifie que la plupart des programmes qui utilisent une imprimante — programmes de traitement de texte, programmes de facturation, etc. — supposent que vous possédez une imprimante Epson. Cela est important à souligner car les autres marques d'imprimante ne sont pas compatibles.

La fiabilité est certainement un facteur important à prendre en compte. Une imprimante peu coûteuse de 2 500 F peut être parfaite pour produire occasionnellement un listage, mais elle ne pourra supporter une utilisation intensive, plusieurs heures d'affilée. De même, le bruit est un facteur souvent négligé. Si vous aimez travailler la nuit, certaines imprimantes ne sont vraiment pas indiquées. L'imprimante a-t-elle un entraînement par friction?

Toutes les imprimantes matricielles possèdent un tracteur à picots qui ne fonctionne qu'avec du papier en continu perforé sur les côtés. Si vous avez besoin d'imprimer sur des feuilles simples, il est nécessaire d'avoir un entraînement par friction.

La dernière question et non la moindre : l'imprimante fonctionnera-t-elle avec votre ordinateur? La plupart des imprimantes matricielles sont livrées avec une prise parallèle Centronics ou avec une interface série RS232.

Détails des caractères
Les exemples d'impression ci-dessous illustrent la différence de qualité que l'on retrouve entre diverses imprimantes matricielles. Ces différences s'établissent principalement en fonction du nombre d'aiguilles utilisées par chaque tête d'impression; celles utilisant le plus grand nombre d'aiguilles produisent les caractères les mieux définis.
Le premier exemple a été produit avec seulement sept aiguilles; ici l'imprimante ne peut produire les jambages des lettres g, p, q et y sous la ligne, on dit donc qu'elle ne produit pas de véritables jambages.
(Cl. Kevin Jones.)



This print out is from a Commodore MPS801. Note the tails of g, p, j and y do not fall below the base of other letters

This print out is from a Mannesmann Tally MT180 which has a nine pin print head. The quality is acceptable.

This print out is from a Tandy DMP-2100 printer which has a 24 pin print head. This gives a good quality result, but at high price.

KEVIN JONES

En formation

Les tableurs ont constitué une des premières applications majeures de l'ordinateur. Leur utilisation « domestique » a cependant été longtemps freinée par leur image professionnelle.

Les tableurs ont de nombreuses caractéristiques inexploitées par leurs utilisateurs. On retrouve une situation équivalente en ce qui concerne les traitements de texte et les bases de données. Il ne faut pas sous-estimer les utilisations domestiques de ces logiciels. Ainsi, les tableurs sont en réalité sources d'idées, même s'ils ont été accaparés par la comptabilité. En réalité, de même que les traitements de texte s'appliquent aux mots, les tableurs peuvent être considérés comme s'appliquant aux concepts. Un tableur est en fait à la fois un éditeur de texte et une calculatrice. Son nom provient du fait qu'il divise l'affichage en lignes et en colonnes comme un tableur de comptabilité, avec des données dans les cases. Ces dernières sont semblables aux carreaux d'une feuille de papier dans la mesure où elles peuvent servir à différents usages. Du texte peut être saisi sur une case, où il demeurera à l'affichage; des données numériques peuvent être entrées pour affichage et calculs. On peut encore y faire figurer des opérations mathématiques sur les constantes. Lorsque les formules mathématiques sont en place, le tableur devient un programme créé par l'utilisateur et prêt à recevoir des données. A chaque saisie de nouvelles données (textes, nombres ou expressions algébriques), les positions sont recalculées tour à tour, permettant au tableur d'être à jour avec les données. Les tableurs peuvent donc être utilisés pour de simples tâches d'édition écran/imprimante, ce qui permet un formatage approprié. En outre, ils réalisent non seulement les calculs que vous auriez faits vous-même (s'ils n'étaient pas si ennuyeux), mais encore ceux que vous n'auriez jamais pensé à effectuer.

Souvent l'utilisation d'un tableur révélera des besoins nouveaux, tels que la tenue d'inventaires, des analyses sportives, l'établissement d'états, la création d'histogrammes de synchronisation « Son et Lumière » pour une représentation théâtrale, l'établissement de la déclaration de revenus, la prise de décisions telles que l'achat ou la location d'un téléviseur, etc. Tout cela pourrait être programmé par un utilisateur averti du BASIC, mais prendrait aussi des heures de développement entre la conception et le dépistage d'erreurs, notamment sur les interminables commandes PRINT TAB, PRINT AT et INPUT, nécessaires au formatage de l'affichage. L'avantage majeur des tableurs est de définir l'affichage en même temps que la relation entre les variables. Cela se fait aussi naturellement que la présentation d'une étude sur une feuille de papier, en écrivant où bon vous semble le texte, les données et les résultats de calculs.

Les tableurs comportent un grand nombre de commandes de présentation. Vous pouvez copier, déplacer ou détruire des cases, insérer ou détruire des lignes ou des colonnes. Vous pouvez aussi définir le format d'une case ou d'un ensemble en termes de taille, justification (alignement avec d'autres éléments de la même colonne), et de position de la virgule. Ce sont précisément ces détails, difficiles à régler avec la plupart des versions du BASIC, et pourtant vitaux pour la présentation et la lisibilité.

Les fonctions de calcul offrent les mêmes ressources. Une seule commande vous permet d'obtenir la moyenne d'une colonne ou d'une ligne, de compter les entrées différentes de zéro, de calculer la somme de tout tableau de valeurs, de trouver les valeurs minimale et maximale d'une liste, et d'utiliser toutes ces commandes dans des opérations mathématiques usuelles du genre « + », « / », SQR (SQuaRe/carré) et ABS (ABSolute value/valeur absolue). Néanmoins tous les tableurs ne disposent pas de ces caractéristiques. Les options possibles dépendent de la mémoire disponible sur votre ordinateur et du prix que vous pouvez mettre dans le logiciel. Les prix à cet égard varient dans un rapport de 1 à 10.

La commande la plus utile d'un tableur est peut être REPLICATE. Cette dernière permet de recopier sur une ou sur plusieurs cases une valeur, de sorte que l'établissement de tableaux récapitulatifs de données — tels que le cumul de taux d'hypothèque d'un mois sur l'autre, ou les dépenses d'une semaine à l'autre — puisse s'obtenir en quelques frappes au clavier. La programmation sur tableur devient rapidement un complément indispensable de l'arithmétique BASIC, rendant possible le calcul rapide d'expressions mathématiques complexes que le BASIC permet difficilement.

Des tableaux entiers peuvent être sauvegardés (SAVE) et chargés (LOAD), depuis un disque ou une cassette. En outre de nombreux tableurs permettent de sauvegarder seulement le texte et les données selon un format de fichier utilisable par un logiciel de traitement de texte et par une base de données. Cela permet d'incorporer en bloc des calculs et des prévisions dans un texte ou un fichier de données. Cela constitue donc un pas appréciable vers le logiciel idéal intégré. Cette possibilité n'est offerte cependant que par les logiciels les plus coûteux.

Le temps des notes

Format
Avec FORMAT, on a fait la largeur de la colonne D, justifié à gauche les cases et affiché les nombres sur 2 décimales.

Copy
Tout bloc de cases peut être copié n'importe où sur le tableau par la commande COPY.

Facteur de correction
Multiplié par la note réelle d'un étudiant, donne la note pondérée.

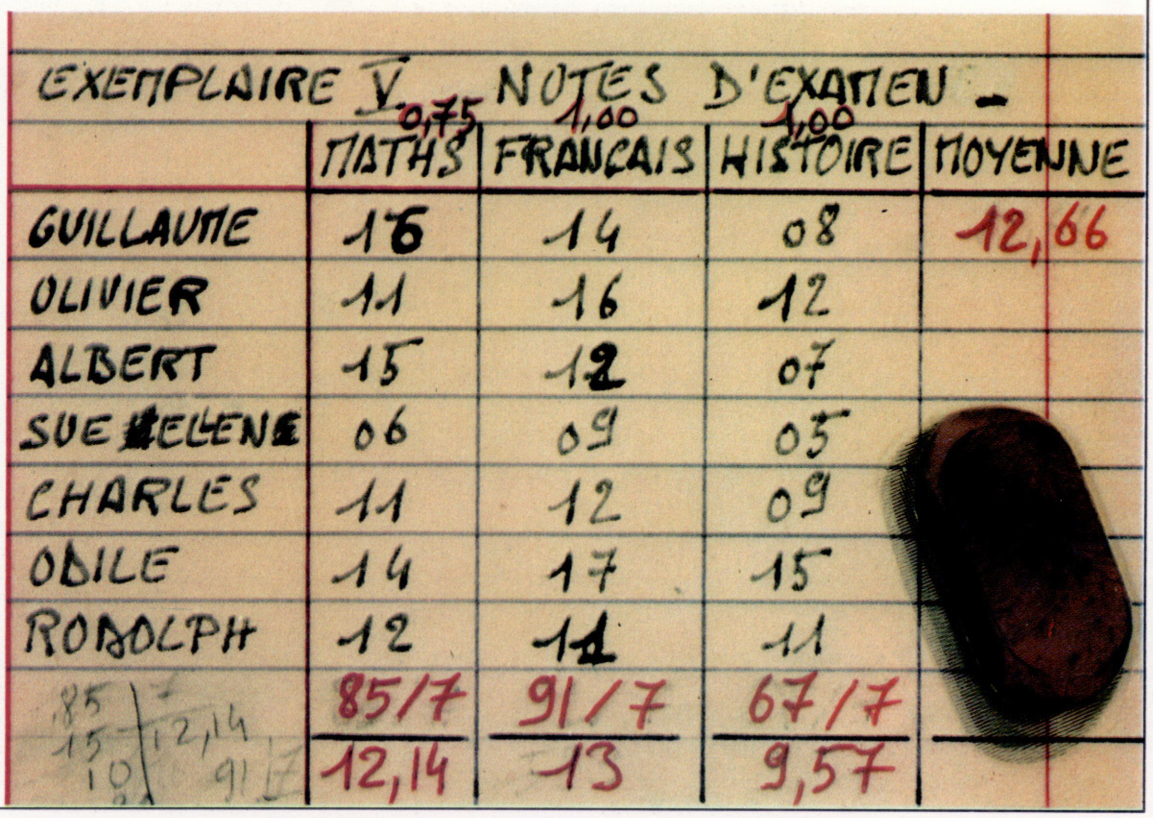
	C	D	F	G	H		J	K	L	M
1	Exemple de facteurs de correction des notes									
2	********************************									
3		notes réelles				*	notes pondérées			*
4	********************************									
5						*	1.5	1	0.5	
6		Maths	Franc.	Hist.	Moy.	*	Maths	Franc.	Hist.	Moy.
7	Guillaume :	16.00	14.00	08.00	12.66	*	24.00	14.00	04.00	14.00
8	Olivier :	11.00	16.00	12.00	13.00	*	16.50	16.00	06.00	12.83
9	Albert :	15.00	12.00	07.00	11.33	*	22.50	12.00	3.50	12.66
10	Suellen :	06.00	09.00	05.00	6.66	*	09.00	09.00	2.50	6.66
11	Charles :	11.00	12.00	09.00	10.66	*	16.50	12.00	4.50	11.00
12	Odile :	14.00	17.00	15.00	15.33	*	21.00	17.00	7.50	15.16
13	Rodolph :	12.00	11.00	11.00	11.33	*	18.00	11.00	5.50	11.50
14		==========	==========	==========	==========	*	==========	==========	==========	==========
15	Moyenne :	12.14	13.00	9.57	11.57	*	18.21	13.00	4.78	11.99
16						*				
17	********************************									
18										

Texte à répéter
Un astérisque figurant dans cette case remplit toute la rangée (commande REPEAT TEXT).

Calc automatique
La commande REPLICATE permet de recopier automatiquement sur d'autres cases une position.

Moyenne
La commande AVERAGE (position#1:position#2) assure le calcul des moyennes.

Afin de comparer les résultats de ses élèves dans différentes matières, un professeur désire obtenir un facteur de correction des résultats d'examen dans chaque matière, afin que la moyenne soit la même d'une matière à l'autre. Il doit pour cela expérimenter divers facteurs de correction pour chaque matière, et calculer les notes à chaque essai. Ce travail fastidieux et sujet à erreur peut être fait bien plus rapidement et plus efficacement par un tableur. Sur le tableau présenté ici, tout est calculé automatiquement sauf les notes réelles. La modification des facteurs de correction par exemple produit en quelques secondes une nouvelle colonne de résultats étalonnés pour la matière concernée.

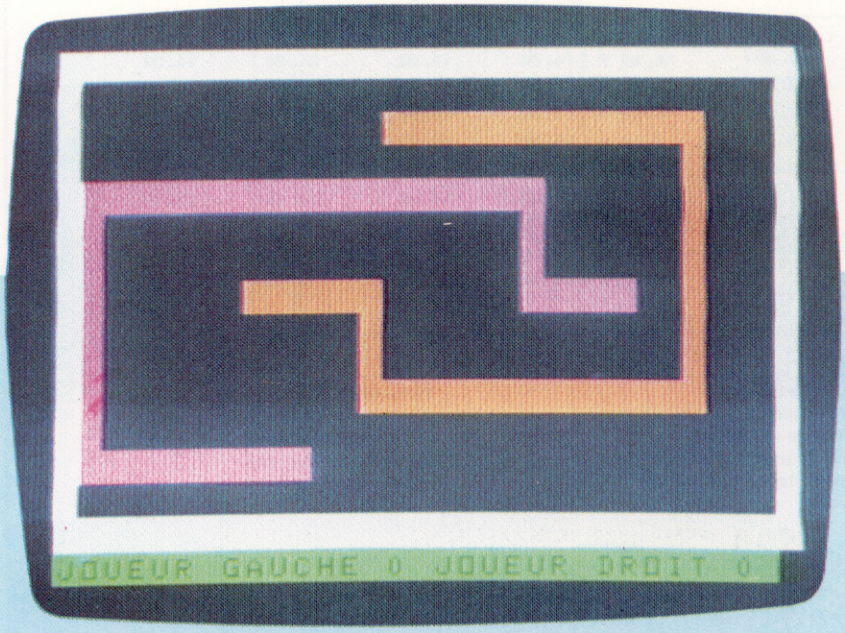


Un tableur doté d'un ensemble satisfaisant de commandes permet des programmes limités seulement par l'imagination des utilisateurs et la taille mémoire disponible.

Les programmes sont d'ailleurs généralement assez longs, et les applications faisant intervenir des caractéristiques de traitement de données puissantes ont rapidement tendance à prendre toute la mémoire. Il convient de tenir également compte du fait que des calculs complexes ralentissent notablement le temps de réponse du programme en calcul.

Trace

Une guerre des tranchées? Non, mais c'est une véritable guerre des lignes que propose Pierre Monsaut dans ce jeu écrit en basic pour le micro-ordinateur Dragon.



Deux joueurs s'affrontent pour se partager l'espace vital. Chacun doit s'efforcer, tout en se déplaçant, de ne jamais recouper sa trace ou celle de son adversaire, et de ne pas sortir du rectangle dessiné sur l'écran. Utilisez les joysticks ou les touches suivantes :

Joueur de droite : P, L, ; et .
Joueur de gauche : W, A, S et Z

```
10 REM *********
20 REM * TRACE *
30 REM *********
40 GOSUB 850
50 GOSUB 630
60 ON JK GOTO 150
70 FOR I=1 TO 50
80 NEXT I
90 D$=INKEY$
100 C1=(D$="L")-(D$=";")+32*((D$
="P")-(D$="."))
110 C2=(D$="A")-(D$="S")+32*((D$
="W")-(D$="Z"))
120 IF C1<>0 THEN D1=C1
130 IF C2<>0 THEN D2=C2
140 GOTO 270
150 K0=JOYSTK(0)
160 K1=JOYSTK(1)
170 K2=JOYSTK(2)
180 K3=JOYSTK(3)
190 IF K0<26 AND K1>37 THEN S1=-
1
200 IF K0>37 AND K1<26 THEN S1=1
210 IF K0 <26 AND K1<26 THEN S1=
-32
220 IF K0>37 AND K1>37 THEN S1=3
2
230 IF K2<26 AND K3>37 THEN S2=-
1
240 IF K2>37 AND K3<26 THEN S2=1
250 IF K2<26 AND K3<26 THEN S2=-
32
260 IF K2>37 AND K3>37 THEN S2=3
2
270 IF S1<>0 THEN D1=S1
280 IF S2<>0 THEN D2=S2
290 P1=P1+D1
300 IF PEEK(P1)<>128 THEN 360
310 POKE P1,J1
320 P2=P2+D2
330 IF PEEK(P2)<>128 THEN 410
340 POKE P2,J2
350 GOTO 60
360 F2=F2+1
370 GOSUB 590
380 IF F2=10 THEN 460
390 D$=INKEY$
400 GOTO 50
410 F1=F1+1
420 GOSUB 590
430 IF F1=10 THEN 500
440 D$=INKEY$
450 GOTO 50
460 CLS
470 PRINT@ 165,"LE JOUEUR GAUCHE
 GAGNE"
480 PRINT@ 202,F2;"A";F1
490 GOTO 530
500 CLS
510 PRINT@ 165,"LE JOUEUR DROIT
GAGNE"
520 PRINT@ 202,F1;"A";F2
530 R$=INKEY$
540 PRINT@ 266,"UNE AUTRE ?"
550 R$=INKEY$
560 IF R$="" THEN 540
570 IF R$<>"N" THEN RUN
580 END
590 FOR I=5 TO 255 STEP 5
600 SOUND I,1
610 NEXT I
620 RETURN
630 CLS 0
640 C=207
650 J1=239
660 J2=255
670 P1=1272
680 P2=1256
690 D1=-1
700 D2=1
710 FOR I=1024 TO 1055
720 POKE I,C
730 POKE I+448,C
740 NEXT I
750 FOR I=1 TO 13
760 POKE I*32+1024,C
770 POKE I*32+1055,C
780 NEXT I
790 PRINT@ 480,"JOUEUR GAUCHE";F
2,"JOUEUR DROIT";F1;
800 POKE P1,J1
810 POKE P2,J2
820 S1=0
830 S2=0
840 RETURN
850 CLS
860 PRINT@ 170,"JOYSTICKS ?"
870 D$=INKEY$
880 IF D$="" THEN 870
890 JK=-(D$="0")
900 RETURN
```

Résultats atteints

Le Colour Genie d'Eaca est une machine robuste conçue pour une utilisation domestique. Son boîtier cache de nombreux composants que l'on retrouve rarement sur des machines de ce prix.

Construit autour du populaire microprocesseur Z80, le Colour Genie possède un excellent clavier de type machine à écrire comptant 62 touches. Parmi ces touches, mentionnons les quatre de fonction, deux touches RESET (qui doivent être pressées simultanément) et une touche Mode Select (Sélection de Mode), qui permet d'utiliser des caractères graphiques prédéfinis à partir du clavier.

La machine a une mémoire de 32 K dont 2 K sont réservés au système. Les graphiques haute résolution utilisent quatre autres K. Les 16 K de ROM renferment une version étendue du BASIC Microsoft, qui n'offre aucune des fonctions de programmation structurée que possèdent la plupart des dialectes BASIC plus récents. Cependant, il autorise l'utilisation de variables entières, de variables de simple ou double précision, de tableaux multi-dimensionnels de tout type de variable, et de fonctions étendues de traitements de chaîne. Il offre plusieurs commandes utiles pour produire des sons et des graphiques haute résolution.

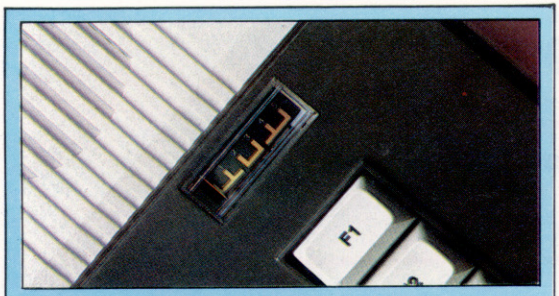
Les fonctions sonores sont relativement évoluées; elles offrent trois canaux (permettant de produire des accords) et la sortie s'effectue par l'intermédiaire du téléviseur. Deux commandes BASIC contrôlent la production de sons (PLAY produit un son prédéfini, tandis que SOUND permet de générer d'autres sons).

Bien qu'étendues et puissantes, les fonctions graphiques du Colour Genie sont maintenant dépassées. L'écran est traité comme deux « pages » (il s'agit en réalité de deux zones de mémoire différentes), l'une de celles-ci stocke et affiche du texte, des blocs de caractères graphiques et des caractères graphiques définis par l'utilisateur, tandis que l'autre page sert à afficher des graphiques haute résolution. Dans le mode texte, le Genie peut afficher jusqu'à 25 lignes de 40 caractères. Dans le mode graphique, la taille de l'affichage est de 160 × 102 points (une faible « haute résolution »).

Manipulation graphique

La touche Mode Select permet d'accéder à la page haute résolution, réservant 4 K de mémoire. Le BASIC offre de nombreuses commandes de manipulation graphique — vous pouvez dessiner des lignes, colorer des zones et définir, dessiner et effacer des formes. Le BASIC comporte, en particulier, des commandes permettant

Un génie modeste
Le Colour Genie est un ordinateur qui n'a jamais atteint une grande réputation. Cette machine a 32 K de mémoire et des manches à balai inhabituels — ils sont livrés en paire assortie avec un clavier numérique intégré pour la somme de 600 F. (Cl. Chris Stevens.)



Indicateur de niveau sonore du Genie
Il peut être très difficile de régler le volume d'une unité à cassette fonctionnant avec un ordinateur; mais le Colour Genie a un compteur de bande pour faciliter la lecture ou l'écriture sur les cassettes.

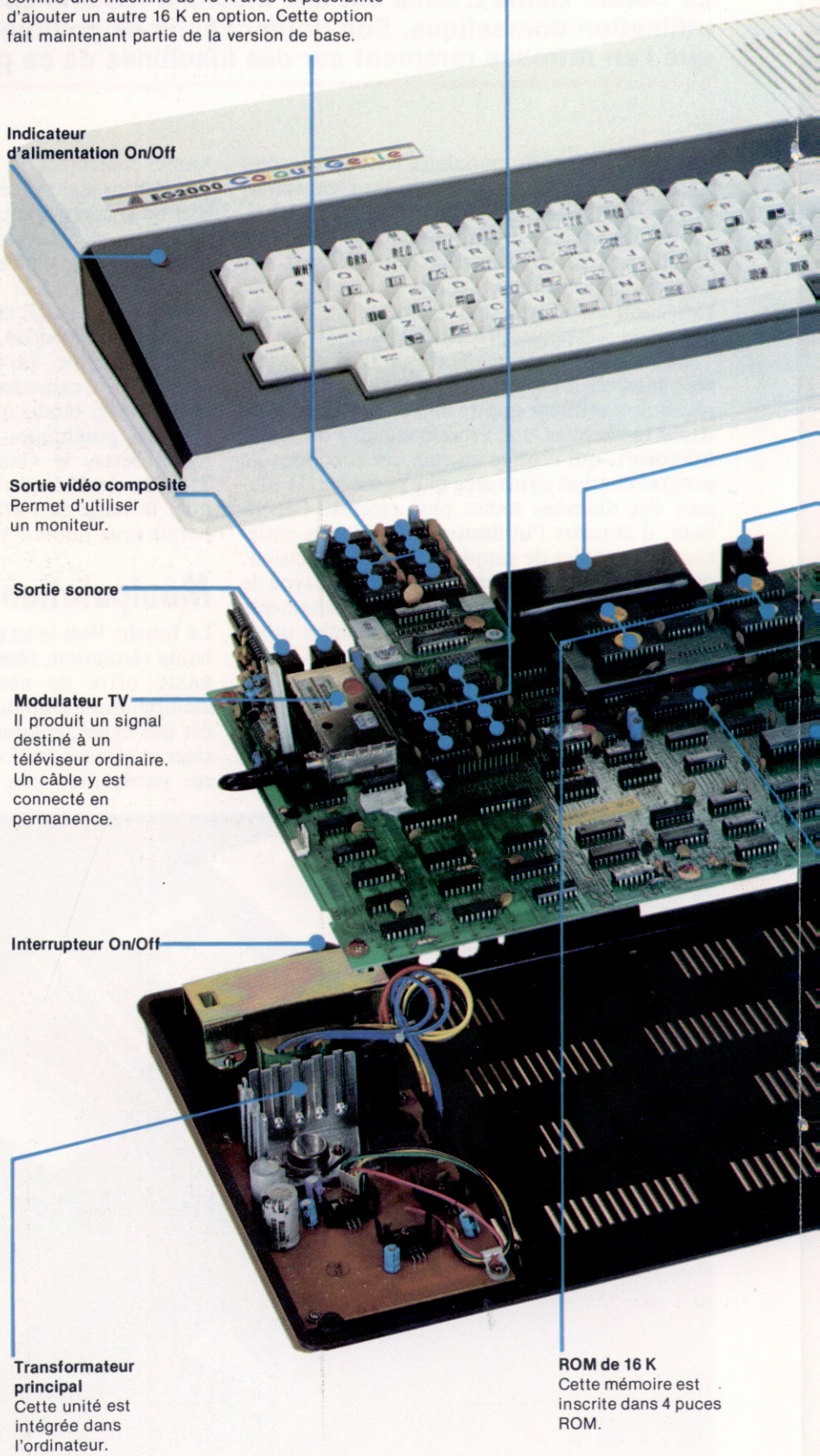
d'effacer la page graphique (FCLS), et de modifier les couleurs de l'arrière-plan (FILL) et du premier plan (FCOLOR). Ce système est moins pratique que la méthode à page unique retenue par la plupart des nouvelles machines, mais il permet de colorer individuellement chaque point. La plupart des jeux de type arcade utilisent l'écran texte pour la vitesse, avec des caractères définis par l'utilisateur pour donner un effet de haute résolution.

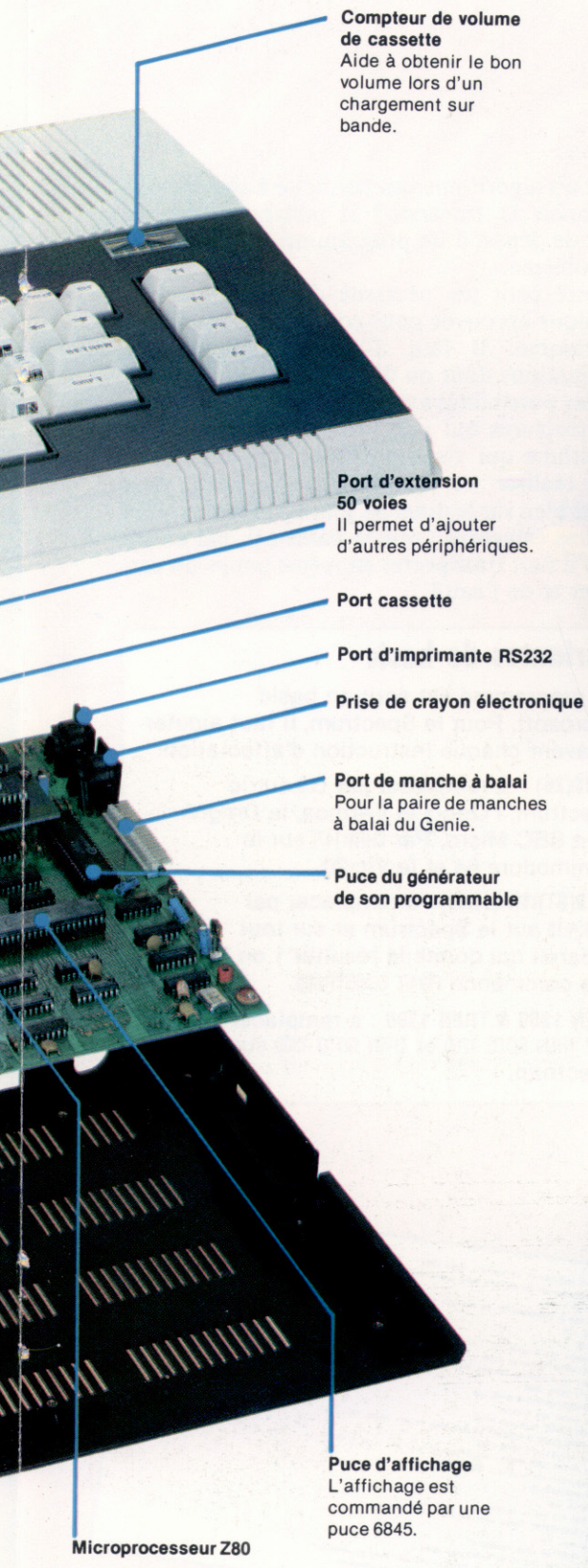
L'affichage est clair et très stable, mais le jeu de caractères utilisés complique un peu la lecture d'un texte. Le Genie offre huit couleurs — blanc, rouge, vert, jaune, cyan, magenta, bleu et orange — toutes ces couleurs pouvant être affichées simultanément sur l'écran texte. Les graphiques haute résolution limitent l'utilisateur à quatre couleurs (rouge, bleu, vert et noir), mais il existe une commande additionnelle (BGRD) pour définir un arrière-plan de page graphique rose.

Plusieurs interfaces sont incluses : un port RS232 pour les imprimantes et pour les modems; un port d'extension servant à connecter des lecteurs de disquettes; une sortie vidéo composite; une sortie audio; une prise de crayon électronique et un port de manche à balai. Parmi les périphériques disponibles, mentionnons un manche à balai, une interface d'imprimante Centronics et des lecteurs de disquettes. Les manches à balai doubles possèdent des commandes intégrées mais sont difficiles à utiliser (il est nécessaire d'exercer une forte pression pour les faire répondre et les manches ne reviennent pas à la position « neutre » centrale lorsqu'ils sont relâchés). Eaca, la société qui fabrique le Colour Genie, ne propose pas de lecteur de disquettes pour la machine.

Un compteur de niveau d'enregistrement est intégré dans le boîtier pour éviter les problèmes de chargement à partir de cassette; l'utilisateur ajuste simplement le volume jusqu'à ce que l'aiguille soit centrée, le chargement devant alors être effectué sans problème. En plus, un « stabiliseur de données » peut être inséré entre le lecteur de cassettes et le conducteur de cassette du Genie; ce système filtre le signal et améliore le fonctionnement de la bande.

Deux manuels accompagnent le Video Genie — un guide du débutant et un manuel BASIC.





Les deux sont clairement écrits, mais peu détaillés, et ils n'ont pas d'index. En fait, le manuel BASIC n'a même pas de table des matières.

Malgré son appparence démodée, le Colour Genie semble offrir un bon rapport qualité/prix. Il se situe sans discussion possible dans la catégorie « domestique » et n'a que peu à offrir à l'utilisateur scientifique ou professionnel. La construction robuste, l'excellent potentiel sonore, la gamme complète de périphériques et un BASIC assez standard peuvent séduire le débutant.

Choix de logiciel
La disponibilité de logiciel pour le Colour Genie est assez limitée, mais la qualité des produits offerts est généralement bonne. La plupart des programmes sont des jeux ; il s'agit souvent de versions modifiées produites pour d'autres machines plus connues. (Cl. Ian McKinnell.)

Colour Genie

PRIX
**

DIMENSIONS
90 × 280 × 340 mm.

UC
Z80; 2,2 MHz.

MÉMOIRE
RAM de 32 K, ROM de 16 K.

ÉCRAN
Jusqu'à 25 lignes de 40 colonnes de texte, graphiques jusqu'à 160 × 102 points, 8 couleurs en mode texte.

INTERFACES
Manches à balai (2), port RS232, port crayon électr.

LANGAGES DISPONIBLES
Basic inclus.

CLAVIER
De type machine à écrire avec 62 touches.

DOCUMENTATION
La machine est livrée avec un manuel pour débutants et un manuel basic. Tous deux sont clairs mais trop brefs pour être d'une quelconque utilité. On peut se procurer un manuel technique.

FORCES
Le Colour Genie est un bon ordinateur « familial ». Il est de construction robuste, utilise le basic Microsoft, offre de bons graphiques et de bons sons, obtenus sur un téléviseur.

FAIBLESSES
La conception technique du Genie est dépassée — son processeur est lent et l'écran est traité en deux « pages ». Peu de programmes sont offerts.

Manches à balai du Colour Genie
Les manches à balai du Colour Genie sont très intéressants mais très chers et difficiles à utiliser. Le manche ne se recentre pas correctement lorsqu'il est relâché.
Les claviers numériques intégrés sont inhabituels mais pas très utiles.

Chris Stevens

Sur le sable!

Conduire un camion à travers le désert n'est pas facile quand on n'a pas assez d'essence... Résoudre le problème demande un certain nombre d'essais, ou même un autre programme!

Notre jeu se déroule au milieu d'un désert de mille kilomètres de large. Tous les cent kilomètres environ, il y a un point de ravitaillement où vous pouvez entreposer des bidons de carburant, ceux-ci sont disponibles à la base principale, qui a largement de quoi satisfaire tous vos besoins. La traversée du désert semble donc chose facile, mais, par manque de place, vous ne pouvez en emporter plus de huit à la fois dans votre camion. Pour réussir, il vous faut donc en déposer à certains points du trajet, puis aller et venir de l'un à l'autre.

De toute évidence, le premier objectif du jeu est de ne jamais manquer de carburant — la base est très loin, et la marche à pied un peu épuisante... Deuxièmement, vous devez effectuer la traversée la plus brève possible, et vous servir d'un minimum de bidons. Le problème est relativement facile quand vous pouvez en emporter huit en une seule fois.

On peut toutefois modifier le jeu de façon à le rendre un peu plus difficile. Que se passe-t-il si votre capacité de transport tombe à quatre ou six bidons? Vous pouvez chercher à le savoir en modifiant la valeur de la variable M à la ligne 60 du programme. Sans doute découvrirez-vous rapidement que si la technique reste la même, les trajets sont plus courts et les aller-retour plus fréquents. Vous est-il possible de mettre au point un algorithme qui permette à chaque fois de réussir la traversée? Il pourrait servir de point de départ à un programme qui *résoudrait* ce problème.

Notre petit jeu nécessite de recourir à une technique éprouvée pour construire ce genre de programme. Il faut d'abord manipuler les informations dont on dispose, essayer de nombreuses possibilités avant de voir émerger certaines structures qui permettent la création d'un algorithme qui viendra à bout des difficultés. Pour réaliser un jeu digne de ce nom, vous devrez bien sûr le doter de graphismes et ajouter d'autres exigences : que se passe-t-il, par exemple, s'il faut transporter en même temps du carburant et de l'eau?

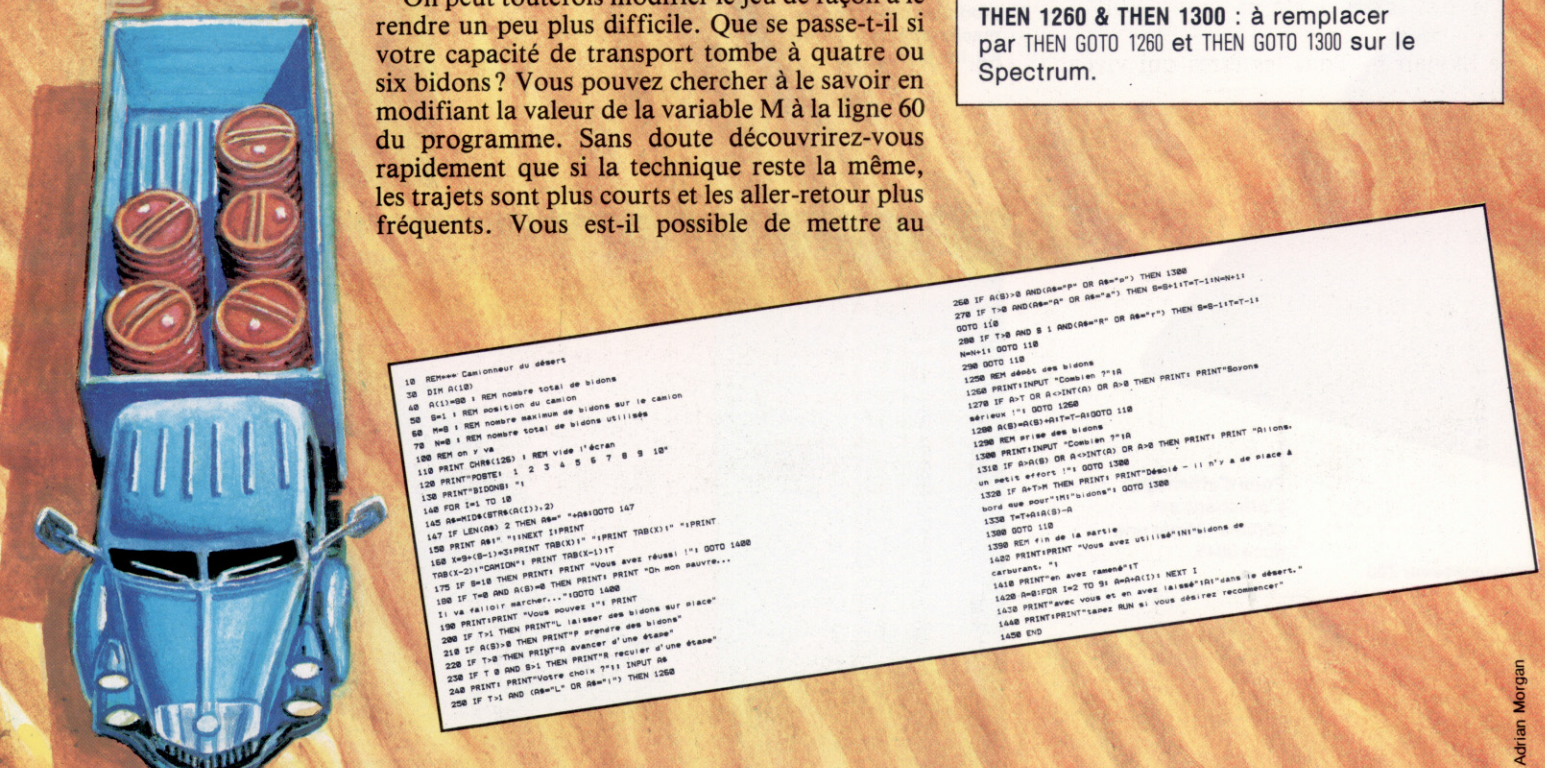
Variantes de basic

Ce programme est écrit en basic Microsoft. Pour le Spectrum, il faut ajouter LET avant chaque instruction d'affectation.

CHR$(26) : à remplacer par CLS sur le Spectrum, l'Oric-1 et l'Atmos, le Dragon et le BBC Micro. Par CHR$(147) sur le Commodore 64 et le Vic-20.

MID$(STR$(A(1),2)) : à remplacer par STR$(A(1)) sur le Spectrum et sur tout appareil qui donne le résultat 1 en réponse à la commande PRINT LEN(STR$(2)).

THEN 1260 & THEN 1300 : à remplacer par THEN GOTO 1260 et THEN GOTO 1300 sur le Spectrum.



```
10  REM=== Camionneur du désert
30  DIM A(10)
40  A(1)=60 : REM nombre total de bidons
50  B=1 : REM position du camion
60  M=8 : REM nombre maximum de bidons utilisés
70  N=0 : REM nombre total de bidons utilisés
100 REM on y va
110 PRINT CHR$(126) : REM vide l'écran
120 PRINT"POSTE: 1  2  3  4  5  6  7  8  9  10"
130 PRINT"BIDONS: ";
140 FOR I=1 TO 10
145 A$=MID$(STR$(A(I)),2)
147 IF LEN(A$) 2 THEN A$=" "+A$:GOTO 147
150 PRINT A$;" ";:NEXT I:PRINT
160 T=3+(B-1)*3:PRINT TAB(X);" ";PRINT TAB(X-1);T
TAB(X-2);"CAMION": PRINT PRINT "Vous avez réussi !": GOTO 1400
170 IF B=10 THEN PRINT: PRINT "Oh non pauvre...
180 IF T=0 AND A(B)=0 THEN PRINT: PRINT "Oh non pauvre...
Il va falloir marcher...":GOTO 1400
190 PRINT:PRINT "Vous pouvez :": PRINT
200 IF T>1 THEN PRINT"L laisser des bidons sur place"
210 IF A(B)>0 THEN PRINT"P prendre des bidons"
220 IF T>0 THEN PRINT"A avancer d'une étape"
230 IF T 0 AND B>1 THEN PRINT"R reculer d'une étape"
240 PRINT: PRINT"Votre choix ?": INPUT A$
250 IF T>1 AND (A$="L" OR A$="l") THEN 1260
260 IF A(B)>0 AND(A$="P" OR A$="p") THEN 1300
270 IF T>0 AND(A$="A" OR A$="a") THEN S=S+1:T=T-1:N=N+1:
GOTO 110
280 IF T>0 AND S 1 AND(A$="R" OR A$="r") THEN S=S-1:T=T-1:
N=N+1: GOTO 110
290 GOTO 110
1250 REM dépôt des bidons
1260 PRINT:INPUT "Combien ?";A
1270 IF A>T OR A<>INT(A) OR A>0 THEN PRINT: PRINT"Soyons
sérieux !": GOTO 1260
1280 A(B)=A(B)+A:T=T-A:GOTO 110
1290 REM prise des bidons
1300 PRINT:INPUT "Combien ?";A
1310 IF A>A1(B) OR A<>INT(A) OR A>0 THEN PRINT: PRINT "Allons,
un petits effort !": GOTO 1300
1320 IF A+T>M THEN PRINT: PRINT"Désolé - il n'y a de place à
bord que pour";M;"bidons": GOTO 1300
1330 T=T+A:A(B)=A
1340 GOTO 110
1390 REM fin de la partie
1400 PRINT:PRINT "Vous avez utilisé";N;"bidons de
carburant. ";
1410 PRINT"en avez ramené";T
1420 A=B:FOR I=2 TO 9: A=A+A(I): NEXT I
1430 PRINT"avec vous et en avez laissé";A;"dans le désert."
1440 PRINT:PRINT"tapez RUN si vous désirez recommencer"
1450 END
```

Adrian Morgan

Une idée creusée à fond

Dû à Matthew Smith, Manic Miner est très vite devenu l'un des plus célèbres jeux pour micro-ordinateurs, grâce à un graphisme soigné et à un humour volontiers macabre.

Manic Miner existe en deux versions : Spectrum 48 K et Commodore 64. Fondamentalement, c'est un jeu très simple inspiré de Kong, un gros succès il y a peu de temps, où il fallait escalader des échelles, des branches et éviter toutes sortes d'obstacles afin de délivrer la malheureuse jeune fille prisonnière de King Kong. Dans Manic Miner, vous jouez le rôle de Willy le Mineur, un prospecteur du centre minier de Surbiton. Il découvre par hasard une mine abandonnée dans laquelle une civilisation oubliée a stocké de l'or et de multiples trésors. Malheureusement, les robots qui travaillaient là sont toujours en activité, et il est très difficile de leur échapper et de ressortir fortune faite. Ils ont la vie dure, ces robots !

Il y a vingt cavernes en tout, et chacune contient quatre clés que Willy doit rassembler pour pouvoir ouvrir la porte qui mène à l'épreuve suivante. Il doit pour cela sauter de corniche en corniche, et certaines s'effondrent sous son poids. Chaque caverne est dotée d'un nom, toujours très allusif : « La Tanière d'Eugène » fait référence à un autre jeune programmeur célèbre, Eugène Evans de chez Imagine : « L'Attaque des téléphones mutants » est une parodie d'un jeu de Jeff Minter de Llamasoft. D'autres endroits s'appellent « Willy rencontre le Roi des Animaux » ou « Baie d'atterrissage de Skylab ». Tous les êtres qui vivent là sont mortels dès qu'on les touche, plantes comprises.

Vous guidez Willy grâce à trois commandes très simples : il peut se déplacer à droite, à gauche, et sauter. C'est un des agréments du jeu : pas besoin de consacrer beaucoup de temps à apprendre les règles; vous pouvez même sélectionner les touches qui vous conviennent le mieux.

Willy a trois vies en tout, mais il ne dispose à chaque fois que d'une quantité d'air limitée; ce qui vous reste d'air est indiqué en bas de l'écran. Après avoir perdu votre dernière vie, vous vous retrouvez dans la première caverne. C'est extrêmement frustrant, et de nombreux joueurs sont parvenus à repartir de tel ou tel endroit, généralement à l'aide d'une série de POKEs bien placés.

La version Commodore n'est rien d'autre qu'une copie très fidèle de celle destinée au Spectrum — l'écran a par exemple été réduit afin d'avoir les mêmes dimensions que l'original. Cela veut dire que les possibilités graphiques et sonores du 64, pourtant bien supérieures, n'ont pas été véritablement mises à profit.

Dans un cas comme dans l'autre, le jeu demeure pourtant passionnant, en raison d'un rythme frénétique et de la difficulté des épreuves, qui doivent être résolues très subtilement. Matthew Smith vient de sortir une suite, Jet Set Willy, qui rencontre le même succès.

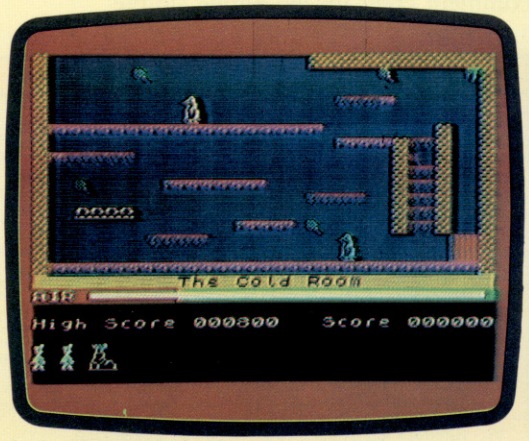
Manic Miner : Spectrum 48 K.
Commodore 64.

Éditeurs : Software Projects,
Bear Brand Complex.

Auteur : Matthew Smith.

Manche à balai : dans les deux versions.

Format : cassette.



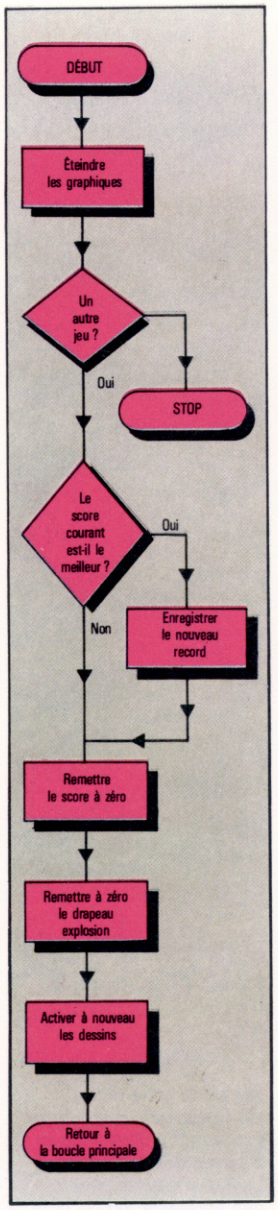
Manic Miner sur le Spectrum

Manic Miner sur le Commodore 64

Six pieds sous terre
L'univers de Manic Miner est rempli d'objets aussi bizarres que merveilleux. L'un des grands plaisirs des amateurs est de faire état de la découverte de telle ou telle chose incroyable au fond des galeries de la mine. (Cl. Ian McKinnell.)

Silence profond

Nous donnons ici la touche finale à notre jeu de chasse sous-marine. Nous allons écrire les routines qui permettent de simuler une explosion lorsque le sous-marin rencontre une mine immergée.



Kevin Jones

Nous avons déjà vu combien il est facile de détecter des collisions entre figures graphiques (ou plan objet) au moyen d'un registre des collisions, V+30. Lorsque cela se produit, le sous-programme COLLISION (qui commence en 5000), se charge de trois tâches. Il doit d'abord représenter à l'écran l'explosion, ensuite augmenter le score du joueur de la valeur correspondante du sous-marin, calculée à partir de sa vitesse (DX), et sa profondeur (Y3). Enfin, le sous-programme ré-initialisera les coordonnées du sous-marin suivant qui devra à nouveau parcourir l'écran. Voyons plus en détail le code du sous-programme COLLISION (lignes 500 à 5250).

La ligne 5010 vide le registre de collision V+30 en y mettant un zéro par POKE. Le manuel du Commodore affirme que le registre de collision de figures graphiques se vide de lui-même lorsque les deux figures se séparent après s'être heurtées. L'expérience montre pourtant que le registre n'est pas toujours ré-initialisé assez vite, ce qui a de curieuses conséquences, telles que des collisions sans causes. La solution sera de vider manuellement le registre. La figure graphique représentant la collision pourra alors être positionnée et allumée lors de la prochaine explosion.

La ligne 5030 donne à l'explosion une coordonnée pour l'axe des X, 10 pixels à droite de celle de la mine immergée. Ce léger décalage positionne l'explosion davantage au-dessus de la mine. X2 prenant sa valeur de la coordonnée X du navire (X0), sa valeur maximale sera de 245. Cela signifie que la valeur maximale pour la coordonnée X de l'explosion est de 255. La coordonnée Y de l'explosion est celle du sous-marin.

La figure graphique de l'explosion est appelée figure 1. La ligne 5040 met à 1 le bit 1 du registre B+21, activant la figure 1 sans affecter la valeur des autres bits du registre. Il est intéressant de noter ici que la figure graphique de l'explosion apparaît au-dessus ou devant les figures du sous-marin et de la mine. C'est ce que l'on appelle la *priorité d'affichage* entre les figures. La règle à cet égard est que les figures de nombre inférieur soient en premier plan par rapport à celles de nombres plus grands. Ce n'est donc pas un hasard si la figure de l'explosion a reçu le numéro 1, et celles de la mine et du sous-marin, respectivement les numéros 2 et 3.

La couleur de la figure graphique de l'explosion est commandée par la position V+40 du composant VIC. Il est possible d'obtenir un effet visuel intéressant en changeant rapidement la couleur de l'explosion par une boucle FOR...NEXT qui introduit successivement des numéros de code-couleur compris entre 1 et 15. Une boucle externe FOR...NEXT permet de répéter ce processus vingt fois (lignes 5060 à 5100). Une fois l'explosion finie, les trois figures graphiques (explosion, mine immergée et sous-marin) doivent disparaître de l'écran. La ligne 5130 inhibe les figures 1, 2 et 3.

Comme nous l'avions dit précédemment, le score du joueur doit être mis à jour par le sous-programme commençant à la ligne 5500. Puisqu'il doit être accru de la valeur du sous-marin (et non pas diminué de cette même valeur, lorsque le sous-marin franchit tout l'écran sans être atteint), la valeur de DS est mise à 1 pour le signaler. En dernier lieu, avant qu'un autre sous-marin puisse traverser l'écran, sa position doit être ré-initialisée par le sous-programme de la ligne 2500, et le dessin du sous-marin à nouveau activé. En outre, le drapeau qui signale le tir d'une mine immergée sera remis à zéro afin de permettre de nouveaux tirs.

Après 3 minutes, le programme quitte la boucle principale et passe à la ligne 400. Lorsque nous avons vu pour la première fois l'utilisation du rythmeur du Commodore 64, la ligne 400 était une instruction END. La routine FIN DE PARTIE permet de refaire un jeu et d'enregistrer les scores. L'organigramme indique les tâches devant y figurer. Elles sont remplies par les lignes 400 à 600 du programme listé. Le code dans ses grandes lignes parle de lui-même, il suffit de savoir que CHR$(19) rapatrie le curseur en haut à gauche de l'écran, et que CHR$(144) affiche en noir (PRINT) ce qui suit. Nous venons de voir comment créer un jeu animé simple

Table des variables utilisées dans le jeu du sous-marin

V	Commencement des registres du composant Vic
FL	Drapeau explosion – mis à 1 lors du tir d'une mine (FLag)
SC	SCore courant du joueur
HS	Le meilleur score (le plus haut)
TI$	Rythmeur propre au Commodore 64 (TImer)
X0	Coordonnée X du navire
X2,Y2	Coordonnées X et Y de la mine
X3,Y3	Coordonnées X et Y du sous-marin
H3,L3	Octets « haut » et « bas » de la coordonnée X du sous-marin (High et Low)
DX	Accroissement en pixels de la coordonnée X du sous-marin
DS	Drapeau indiquant si le score doit être augmenté (DS = 1), ou diminué (DS = – 1)

Le jeu du sous-marin - listage final

```
10  REM ******************************
30  REM ** PROJET DE PROGRAMMATION *
      ** SUR COM 64 ***************
70  REM ******************************
90  POKE55,0:POKE56,48:CLR:
REM POSITION MEMOIRE HAUTE DU PLUS
BAS REGISTRE
100 V=53248:FL=0:SC=0
110 GOSUB1000: REM INITIALISATION DE
L'ECRAN
120 GOSUB2000: REM CREATION FIGURE
130 GOSUB2500: REM INITIALISATION DES
COORDONNEES
140 TI$="000000"
200 REM **** BOUCLE PRINCIPALE ****
210 REM   ** RYTHMEUR **
220 PRINTCHR$(19)::PRINTTAB(14)CHR$(5):
"TIME "MID$(TI$,3,2)":"RIGHT$(TI$,2)
225 IFVAL(TI$)>259 THEN 400:REM FIN DE
PARTIE
230 GET A$
240 IF A$="Z" THEN XO=XO-1.5:IF XO>24
THENXO=24
250 IF A$="X" THEN XO=XO+1.5:IF XO>245
THENXO=245
260 IF A$="M" AND FL=0 THEN GOSUB3000:
REM INITIALISATION TIR
270 REM  ** DEPLACEMENT NAVIRE  **
290 POKE V,XO
300 REM  ** DEPLACEMENT SOUS-MARIN **
310 X3=X3+DX
320 REM** SI LE SOUS-MARIN ATTEINT LE
BORD DE L'ECRAN **
330 IF X3>360 THEN DS=-1:GOSUB5500:
GOSUB2500
340 H3=INT(X3/256):L3=X3-256*H3
350 POKE V+6,L3
360 IF H3=1/THEN POKE V+16,PEEK(V+16)OR8:
GOTO380
370 POKE V+16,PEEK(V+16)AND247
380 IF FL=1 THEN GOSUB4000:
REM TRAJECTOIRE DU TIR
390 GOTO 200:REM REPRENDRE LA BOUCLE
PRINCIPALE
400 REM **** FIN DES CONDITIONS DE
JEU ****
410 REM   ** INHIBER LES DESSINS **
420 POKE  V+21,0
430 REM   ** RE-INITIALISER LES
COORDONNEES DU NAVIRE ET DU SOUS-MARIN **
440 XO=160:GOSUB 2500
450 INPUT" "UN AUTRE JEU (O/N)";AN$
460 IF AN$<>"Y"THENEND
480 REM   ** MESSAGE D'EFFACEMENT **
490 PRINT CHR$(19)::REM RETOUR CURSEUR
500 FOR I=1 TO 120
510 PRINT" ";
520 NEXT I
540 REM   ** INITIALISER LE MEILLEUR
SCORE **
550 IF SC>HS THEN HS=SC
560 PRINT CHR$(19);CHR$(144);" SCORE 000":
SC=0
570 PRINT CHR$(19);
580 PRINT TAB(26);CHR$(144);"HI SCORE";HS
600 REM   ** RE-INITIALISER LE RYTHMEUR
ET LE DRAPEAU
610 TI$="000000":FL=0
630 REM   ** ACTIVER SOUS-MARIN ET NAVIRE **
640 POKE V+21,9
650 GOTO200: REM REPRENDRE LA BOUCLE
1000 REM **** INITIALISATION DE L'ECRAN
1010 PRINT CHR$(147):REM EFFACEMENT DE L'ECRAN
1030 REM   ** COULEUR DE LA MER **
1040 POKE 53281,14:POKE 53280,6
1050 FOR I=1264 TO 1943
1060 POKE I,160:POKE I+54272,6
1070 NEXT
1090 REM   ** FOND DE LA MER **
1100 FORI=1944 TO 2023
1110 POKE I,102:POKE I+54272,9
1120 NEXT
1130 POKE 650,128:REM REPRENDRE LES VARIABLES
CLEFS
1150 REM   ** SCORE **
1160 PRINT CHR$(19);CHR$(144);" SCORE 000"
;SPC(16);"HI SCORE 000"
1170 RETURN
2000 REM **** CREATION FIGURES ****
2020 REM   ** LIRE DONNEES NAVIRE ****
2030 FOR I=12288 TO 12350
2040 READ A:POKE I,A:NEXT I
2060 REM   ** LIRE DONNEES SOUS-MARIN **
2070 FOR I=12352 TO 12414
2080 READ A:POKE I,A:NEXT
2100 REM   ** LIRE DONNEES MINE **
2110 FOR I = 12416 TO 12478
2120 READ A:POKE I,A:NEXT
2140 REM   ** LIRE DONNEES EXPLOSION **
2150 FOR I = 12480 TO 12542
2160 READ A:POKE I,A:NEXT
2180 REM   ** AFFECTER LES POINTEURS **
2190 POKE 2040,192:POKE 2041,193:POKE 2042
,194
2200 POKE2043,195
2220 REM   ** AFFECTER LES COULEURS **
2230 POKE V+39,0:POKE V+40,1:POKE V+41,0
2240 POKE V+42,0
2260 REM   ** AFFECTER LES COORDONNEES INIT. **
2270 POKE V+1,80:X0=160: REM COORDONNEES NAVIRE
2280 POKE V+29,15:POKE V+23,2
2300 REM   ** ACTIVER FIGURES 0 ET 3
2310 POKE V+21,9
2320 RETURN
2500 REM **** RE-INITIALISER COORDONNEES NAV. ****
2510 Y3=110+INT(RND(TI)*105)
2520 POKE V+7,Y3:POKE V+6,0
2530 X3=0:DX=RND(TI)*3+1
2540 POKE V+16,0
2550 RETURN
3000 REM **** INITIALISATION TIR ****
3020 REM   ** METTRE LE DRAPEAU **
3030 FL=1
3050 REM   ** AFFECTER LES COORDONNEES **
3060 Y2=95:X2=XO
3070 POKE V+4,X2:POKE V+5,Y2
3090 REM   ** ACTIVER LA FIGURE N°2
3100 POKE V+21,PEEK(V+21)OP4
3110 RETURN
4000 REM **** TRAJECTOIRE DE LA MINE ****
4020 REM   ** DIMINUER LA COORDONNEE Y **
4030 Y2=Y2+2
4050 REM   ** TESTER LA PROFONDEUR § ARRET **
4060 IF Y2>Y3+25 OR Y2>216 THEN POKEV+21,
PEEK(V+21)AND251:FL=0
4070 POKE V+5,Y2
4090 REM   ** TEST DE COLLISION **
4100 IF PEEK(V+30)=12 THEN GOSUB 5000:
REM ROUTINE DE COLLISION
4110 RETURN
5000 REM **** ROUTINE DE COLLISION ****
5010 POKE V+30,0:REM CLR REGISTRE DE COLLISION.
5020 REM   ** ACTIVER LA FIGURE DE COLLISION **
5030 POKE V+2,X2+10:POKE V+3,Y3
5040 POKE V+21,PEEK(V+21)OR2
5060 REM   ** COULEURS DE CLIGNOTEMENT **
5070 FOR I=1 TO 20
5080 FOR J=1TO 15
5090 POKE V+40,J
5100 NEXT J:NEXT I
5120 REM   ** INHIBER LES DESSINS 1,2 & 3**
5130 POKE V+21,PEEK(V+2)AND241
5150 REM   ** MISE A JOUR DU SCORE **
5160 DS=1:GOSUB 5500
5180 REM   ** RE-INITIALISATION DES
COORDONNEES DU SOUS-MARIN ET DU DRAPEAU
5190 FL=0:GOSUB 2500
5210 REM   ** TURN SUB BACK ON **
5220 POKE V+21,PEEK(V+21)OR8
5230 RETURN
5500 REM **** MISE A JOUR DU SCORE ****
5510 SC=SC+INT(Y3+DX*30)*DS
5520 IF SC<0 THEN SC=0
5530 PRINT CHR$(19);CHR$(144)" SCORE";SC;
CHR$(157);"
5540 RETURN
6000 REM **** DONNEES NAVIRE ****
6010 DATA0,0,0,0,0,0,0,0,0
6020 DATA0,128,0,0,192,0,0,192,0
6030 DATA0,192,0,1,224,0,1,224,0
6040 DATA13,224,0,3,248,128,3,253,8
6050 DATA15,254,16,31,255,48,255,255,255
6060 DATA127,255,254,63,255,254,31,255,252
6070 DATA0,0,0,0,0,0,0,0,0
6100 REM **** DONNEES DE L'EXPLOSION
6110 DATA0,0,0,0,0,0,0,16,0,0,8,0,4,16
6120 DATA0,3,2,64,1,56,128,12,255,144
6130 DATA1,238,40,5,151,0,11,121,0,1
6140 DATA183,0,25,214,96,0,236,48,6,24
6150 DATA152,3,98,0,8,51,0,0,96,128,0
6160 DATA64,0,0,0,0,0,0,0
6170 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6200 REM **** DONNEES MINES ****
6210 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6220 DATA0,0,0,32,0,0,32,0,0,32,0,0,32,0
6230 DATA0,0,0,0,0,0,0
6240 DATA2,0,0,2,0,0,2,0,0,2,0,0
6250 DATA0,0,0,0,0,0,0,0
6260 DATA0,0,0,0,0,0,0,0
6300 REM **** DONNEES DU SOUS-MARIN ****
6310 DATA0,0,0,0,0,0,0,0,0,0,0,0,0
6320 DATA0,8,0,0,12,0,0,12,0
6330 DATA0,12,0,0,28,0,0,60,0
6340 DATA0,126,0,199,255,255
6350 DATA239,255,255,127,255,255
6360 DATA255,255,254,199,255,254
6370 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Voici le listage final de notre programme de jeu ainsi qu'une table de ses variables clefs.
Le listage comprend de nombreuses déclarations REM (de commentaire), qui aident à la compréhension du programme. Vous pouvez ne pas les entrer dans votre ordinateur lorsque vous saisirez le programme. Cependant, il vous faudra veiller à ne pas supprimer une ligne REM nécessaire à une autre partie du programme. Ainsi si vous décidez de supprimer la ligne REM 400, et que ce numéro de ligne est utilisé dans une instruction GOTO à la ligne 225, vous obtiendrez un message "UNDEFINED STATEMENT ERROR AT LINE 225" (Erreur - instruction non définie à la ligne 225) et le programme s'interrompra. Le meilleur moyen d'éviter cela est de ne supprimer que les commentaires REM intervenant en fin de ligne, et les lignes utilisant le signe ":" pour espacer le code.

La division

Nous terminons cette série par une brève étude de la division binaire sans signe et l'usage de routines ROM du système d'exploitation pour programmer l'affichage écran en langage d'assemblage.

Nous avions utilisé la longue méthode manuelle de multiplication comme algorithme pour la multiplication binaire. La même méthode manuelle nous servira de modèle pour la division binaire. Considérons la division binaire suivante :

```
     00001110 r00          quotient
1011)10011010              / dividende
   - 1011                  diviseur soustrait
     10000
   - 1011                  diviseur soustrait
     1011
   - 1011                  diviseur soustrait
     00                    pas de reste
```

Le principe consiste à soustraire plusieurs fois le diviseur des bits d'ordre supérieur du dividende. Selon le résultat de cette soustraction, on décale un 0 ou un 1 dans le quotient. Le reste est le résultat de la dernière soustraction d'un diviseur.

Division de 16 bits par 8 bits

Z80			6502		
START	LD	A,(DIVSR)	START	LDA	#$00
	LD	D,A		STA	QUOT
	LD	E,$00		STA	RMNDR
	LD	HL,(DVDND)		LDX	#$08
	LD	B,$08		LDA	DVDHI
LOOP0	AND	A		SEC	
	SBC	HL,DE		SBC	DIVSR
	INC	HL	LOOP0	PHP	
	JP	P,POSRES		ROL	QUOT
NEGRES	ADD	HL,DE		ASL	DVDLO
	DEC	HL		ROL	A
POSRES	ADD	HL,HL		PLP	
	DJNZ	LOOP0		BCC	CONT1
	LD	(QUOT),HL		SBC	DIVSR
	RET			JMP	CONT2
DIVSR	DB	$F9	CONT1	ADC	DIVSR
DVDND	DW	$FDE8	CONT2	DEX	
QUOT	DB	$00		BNE	LOOP0
RMNDR	DB	$00		BCS	EXIT
				ADC	DIVSR
				CLC	
			EXIT	ROL	QUOT
				STA	RMNDR
				RTS	
			DIVSR	DB	$F9
			DVDLO	DB	$E8
			DVDHI	DB	$FD
			QUOT	DB	$00
			RMNDR	DB	$00

Les moyens qui mettent en œuvre cet algorithme en langage d'assemblage ne sont pas aussi évidents que pour la multiplication. Comme précédemment, la version Z80 utilise la puissance et la souplesse de ses registres à 16 bits, alors que le 6502 doit se débrouiller avec 8 bits. Le diviseur est dans une adresse désignée par DIVSR, le dividende dans DVDND, le quotient dans QUOT, et le reste dans RMNDR.

Notez que dans les deux cas, lorsque le diviseur est soustrait du dividende partiel avec un résultat négatif, il faut le rétablir en y rajoutant le diviseur. La version 6502 est remarquable pour son traitement de registre d'état du processeur (PSR) après la soustraction du diviseur : le drapeau de retenue doit être permuté dans le quotient, mais son état doit aussi être conservé pour indiquer le résultat de la soustraction. Par conséquent, le PSR est entré sur la pile avant la permutation, et ressorti après, rétablissant ainsi la retenue dans son état d'après soustraction. Nous avons maintenant examiné les quatre règles de l'arithmétique.

Sortie sur écran

Jusqu'ici nous avons utilisé la mémoire RAM et l'UC comme un système pour calculer, et nous avons laissé les résultats de nos efforts quelque part en RAM pour les inspecter manuellement en utilisant un programme moniteur. Cela n'est évidemment pas satisfaisant; mais tant que nous considérions l'arithmétique et les appels de sous-programmes, les sorties sur écran à partir du langage machine n'avaient pas leur place.

La plupart des micros disposent d'un affichage mémoire. Cela signifie qu'une zone de RAM est réservée au stockage d'une image de l'écran. L'affichage sur écran est composé de points, ou pixels, qui sont allumés ou éteints. Ils peuvent donc être représentés par des uns (allumé) ou des zéros (éteint) binaires, et tout le contenu de l'écran peut être considéré comme une « correspondance » entre les points et les bits qui comprennent les octets de la RAM écran. Malheureusement, bien que le BBC Micro, le Spectrum et le Commodore 64 utilisent tous cette technique de correspondance, aucun d'eux ne le fait directement. Pour nous, la méthode la plus simple serait de diviser chaque ligne d'écran en octets de pixels numérotés consécutivement de gauche à droite (l'octet le plus à gauche d'une ligne suivant le plus à droite de la ligne précédente).

L'écran du Spectrum est toujours en mode haute résolution, et une zone fixée de mémoire est réservée pour la représentation d'écran. Cette correspondance est cependant complexe, car l'écran est divisé horizontalement en trois blocs de huit lignes PRINT, dont chacune est divisée horizontalement en huit lignes de pixels. L'adressage des octets comprenant ces lignes est séquentiel à l'intérieur des lignes, mais pas entre les lignes. Le BBC Micro et le Commodore 64 ne sont pas plus simples. Pour l'instant, c'est bien plus facile à comprendre si nous nous limitons à la sortie de caractères ASCII à l'écran.

Voilà quelque chose que la machine fait tout le temps, et c'est pourquoi il existe des routines en langage machine en ROM à cet effet. Étant donné une description suffisamment détaillée de leur fonctionnement, nous pouvons appeler ces routines à partir de nos propres programmes en langage d'assemblage. Ce qu'il nous faut savoir, ce sont l'adresse d'appel, les registres de communication et tous les préliminaires nécessaires.

Sur le Spectrum, il n'y a pas de préliminaires à observer, et le registre de communication est l'accumulateur qui doit contenir le code ASCII du caractère à imprimer. Nous n'avons besoin que de sortir l'instruction RST $10, et le caractère dont le code se trouve dans l'accumulateur sera imprimé à l'écran à l'emplacement du curseur. C'est à peu près le même principe que pour les deux autres systèmes, mais l'opc RST (ReSTart = repartir) est particulier au jeu de commandes de Z80 : c'est une instruction de branchement en page zéro à un octet qui doit prendre l'un des huit seuls facteurs possibles — $00,$08,$10,$18, etc., jusqu'à $38. Chacune de ces positions indique l'adresse de départ d'une routine ROM, quelque part en page zéro. Ces routines sont typiquement réservées aux entrées et sorties, et on les appelle par l'instruction RST plutôt que directement par l'adresse. Cela en partie pour la rapidité (il est plus rapide d'utiliser RST que CALL, quoique seule l'UC remarque la différence), et en partie à cause de la portabilité du programme. Si tous les programmeurs de Z80 savent que RST $10 appelle la routine PRINT sur les machines Z80, alors il n'y a pas à se soucier de l'endroit où les concepteurs de logiciels ont effectivement placé la routine PRINT, et ceux-ci sont libres de la mettre n'importe où, à condition d'arranger la page zéro de façon que les emplacements RST dirigent les programmes aux adresses de départ des routines communes.

Sur le BBC Micro la procédure est semblable : la combinaison d'un code ASCII dans l'accumulateur avec une commande JSR $FFEE aura pour effet d'imprimer le caractère sur l'écran à l'emplacement du curseur. C'est la routine OSWRCH, souvent mentionnée dans la littérature se rapportant au BBC.

Le Commodore 64 suit le modèle des deux autres machines. Un code ASCII dans l'accumulateur et une commande JSR $FFD2 produit l'impression du caractère à l'emplacement du curseur. C'est la routine CHKOUT.

Voilà le principe général d'utilisation des routines ROM et des registres de communication. Une communication entre le programme appelant et un sous-programme peut se faire des deux manières — une routine d'entrée, par exemple, peut faire passer un caractère d'un dispositif externe à l'UC via l'accumulateur. Même si aucune information essentielle n'est passée de cette manière, une erreur de code peut fort bien résulter du sous-programme par l'un des registres. Il existe une documentation à ce sujet pour les différentes machines.

L'entrée à partir du clavier et d'autres dispositifs sera traitée ultérieurement, de même que les tracés en haute résolution à partir du langage machine. Nous concluons cet article par une récapitulation des divers aspects de la programmation en langage d'assemblage et machine.

Récapitulation

Nous avons commencé ce cours en survolant le langage machine d'un point de vue général, en essayant de le démystifier quelque peu et de le placer dans son contexte comme une espèce de langage parmi d'autres, que nous (et les ordinateurs) utilisons. Nous avons vu comment la même séquence d'octets en RAM peut être interprétée tantôt comme une chaîne de données ASCII, tantôt comme une ligne de programme BASIC, tantôt comme une chaîne d'adresses à deux octets, ou encore comme une séquence d'instructions langage machine. Quelques minutes passées à jouer avec un programme moniteur en langage machine devraient vous convaincre que certaines séquences d'octets peuvent être décomposées en trois séquences d'instructions, toutes différentes, mais également valides — selon que l'on commence la décomposition au premier, deuxième ou troisième octet de la séquence. Rien d'intrinsèque au langage ne peut empêcher cela, et l'UC elle-même ne peut dire si elle exécute le code que vous avez écrit, ou une « salade » transposée accidentellement dans la mémoire.

Nous avons ensuite considéré l'organisation de la mémoire et les conventions d'adressage. Pour cela, nous avons dû commencer l'étude de l'arithmétique binaire pour avoir une vue d'ensemble de l'UC — dans les processeurs huit bits, nous sommes limités, sauf cas particuliers, à un octet (autrement dit, à l'ensemble des nombres décimaux compris entre 0 et 255). Après avoir constaté la pertinence de l'arithmétique binaire, les limitations du système décimal quand il s'agissait de langage d'assemblage sont clairement apparues. En explorant l'idée de mémoire paginée, nous avons vu comment la taille des pages logiques devait dépendre de la base de numération et, dans un système binaire, cela implique que la taille des pages doit être une puissance de deux. Deux à la puissance huit donne 256 — nombre magique dans un système à microprocesseur huit bits.

Le binaire est très vite devenu trop peu maniable et disposait trop à l'erreur ; nous som-

mes donc passés au système de numération hexadécimal (basé sur le nombre 16). Nous avons vu qu'un octet binaire peut être entièrement représenté par deux chiffres hex, de $00 à $FF, un chiffre représentant l'état des quatre bits les moins significatifs, l'autre les quatre plus significatifs de l'octet.

La façon dont les programmes BASIC sont stockés dans la zone de programme a été totalement examinée. En écrivant le codage des mots clés comme une autre forme de langage machine, nous avons eu un aperçu du système d'exploitation. Notre discussion des marqueurs de fin de ligne a montré comment l'interpréteur BASIC peut savoir où se termine une partie de code et où commence une autre, et l'adressage de lieu du Comodore a introduit à la fois la convention d'adresse lo-hi et l'idée d'adressage indirect.

De là, nous sommes entrés directement dans le langage d'assemblage proprement dit. Nous avons commencé par les opérations primitives de l'UC commandées par les codes opérations (opc) à huit bits qui constituent ses instructions de programme. En explorant à fond l'idée de codage, il n'y avait qu'un pas jusqu'aux mnémoniques de langage d'assemblage. Ce pas franchi, il devint évident que la programmation en langage machine, en assemblage ou en BASIC n'était autre que de la programmation, et que ce qui comptait était de résoudre le problème logique avant de se soucier de la façon de coder la solution. La résolution d'un problème était le thème central du cours. Mais l'obscurité de certains concepts de langage d'assemblage nous a incités à dissiper d'abord la confusion qui assaille la plupart des gens au premier contact avec des langages peu évolués.

Nous nous sommes ensuite arrêtés aux pratiques de chargement et de mise en route des programmes en langage machine sur des ordinateurs qui étaient plus ou moins prévus pour des programmes en BASIC. Nous avons examiné les variables système et les pointeurs du système d'exploitation sur le BBC Micro, le Spectrum et le Commodore 64, et avons appris comment « voler » de l'espace au BASIC.

Nous avons jeté un coup d'œil à l'architecture des petits systèmes informatiques et aux UC Z80 et 6502, puis nous avons commencé à écrire des programmes en langage d'assemblage pour manipuler la mémoire et l'accumulateur. Les pseudo-instructions d'assembleur furent alors introduites, constituant un pas vers la pratique et la réalité, mais également un autre plus loin du langage machine, de l'assemblage manuel et de la tâche fastidieuse de la programmation non évoluée.

A ce stade, la nécessité d'une construction logique d'un langage de programmation était évidente, et nous avons considéré le registre d'état du processeur (PSR). Son rôle d'enregistreur des résultats d'opérations de l'UC fut immédiatement illustré dans une introduction à l'arithmétique binaire, utilisant l'instruction d'« addition avec retenue ». Le rôle central du PSR et, en arithmétique, du drapeau de retenue, était alors évident. Nous nous sommes donc concentrés sur le PSR et les instructions associées.

Nous avons examiné brièvement les différents modes d'adressage; on a donné le plus d'attention à l'adressage indexé à cause de son importance dans les boucles, les listes et les tableaux. Une fois introduites ces structures, la nécessité est apparue d'avoir une classe d'instructions pour modifier le déroulement du contrôle dans un programme, et nous avons donc commencé à regarder les instructions de branchement conditionnel tout en explorant encore l'adressage indexé et indirect. Avec le branchement conditionnel, l'arithmétique primitive et les structures de type tableau, nous avons presque tous les éléments de base d'un langage de programmation, quel qu'il soit. Il ne reste plus qu'à les revêtir de formes par la pratique et l'investigation systématique.

On a étudié l'appel et le retour d'un sous-programme en langage d'assemblage, à la fois pour eux-mêmes et comme moyen d'introduire la dernière zone inexplorée du système d'exploitation : la pile. Son fonctionnement et son utilisation ont introduit de nouvelles démarches au répertoire de la programmation en langage machine, tandis qu'une étude plus approfondie des registres d'UC et de leurs interactions introduisait de nouvelles possibilités dans la manipulation de la mémoire et du microprocesseur.

Finalement, en connaissant l'architecture du microprocesseur et le vocabulaire des instructions d'opc, nous avons approché l'arithmétique binaire. Les particularités de la soustraction, la complexité de la multiplication et de la division ont toutes été vues en détail. Il nous reste à examiner les astuces de la programmation en langage machine en étudiant et en explorant les tâches spécifiques aux processeurs sur lesquels nous avons concentré notre attention dès le début (le Z80 et le 6502), ainsi que d'autres processeurs, tel le 6809 utilisé par les Dragon 32 et 64.

Solutions aux exercices précédents

1. La solution la plus rapide est certainement une routine écrite spécifiquement pour des multiplicandes à 16 bits, sur les mêmes lignes que la routine 8 bits du dernier numéro. Par ailleurs, si l'on partage la multiplication 16 bits en deux multiplications séparées à 8 bits (multiplicateur par octet lo, suivi de multiplicateur par octet hi), alors vous appelez deux fois la routine à huit bits, ajustez la retenue provenant de l'octet lo, et stockez les résultats dans les octets du produit.

2. Une multiplication utilisant l'addition répétée consiste simplement en une boucle dont le compteur est la valeur du multiplicateur; chaque fois que la boucle est exécutée, le multiplicande est additionné au produit.

ROR — PERMUTER A DROITE — 6502

Registre – **6A** (1 octet)

Le contenu de l'octet est permuté d'un bit à droite en passant par la retenue.

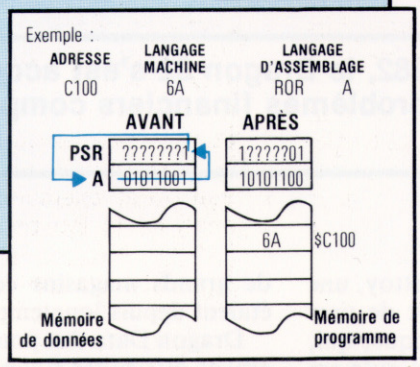
EFFET SUR LE PSR

	S	V		B	D	I	Z	C
MSB	X						X	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	6A	ROR A



RR — PERMUTER A DROITE — Z80

Registre – **CB** (2 octets)

Le contenu de l'octet est permuté d'un bit à droite en passant par la retenue.

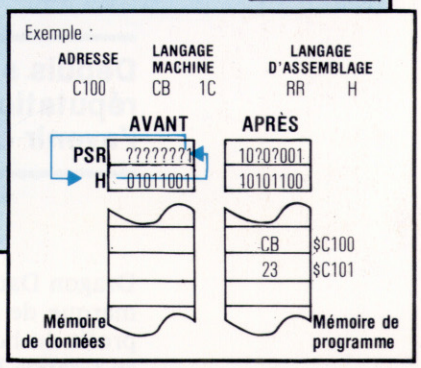
EFFET SUR LE PSR

	S	Z	H		V	N	C
MSB	X	X	0		X	0	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	CB 1C	RR H



SBC — SOUSTRAIRE AVEC RETENUE — 6502

Absolu – **ED** (3 octets)

Le contenu de l'adresse mémoire est soustrait de l'accumulateur ; le drapeau montre l'état de la retenue.

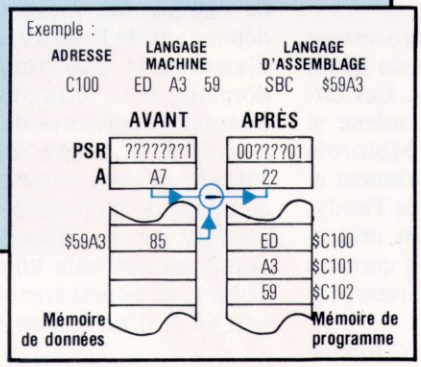
EFFET SUR LE PSR

	S	V		B	D	I	Z	C
MSB	X	X					X	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	ED A3 59	SBC $59A3



SBC — SOUSTRAIRE AVEC RETENUE — Z80

Immédiat – **DE** (2 octets)

Le contenu de l'octet suivant les opc est soustrait de l'accumulateur.

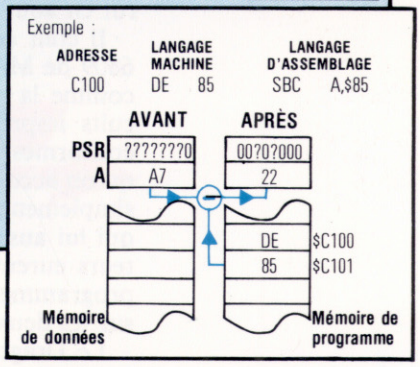
EFFET SUR LE PSR

	S	Z	H		V	N	C
MSB	X	X	X		X	X	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	DE 85	SBC A,$85



ASL — DÉCALER A GAUCHE — 6502

Registre – **0A** (1 octet)

Le contenu de l'octet est décalé d'un bit à gauche en passant par la retenue ; le zéro est décalé dans lsb.

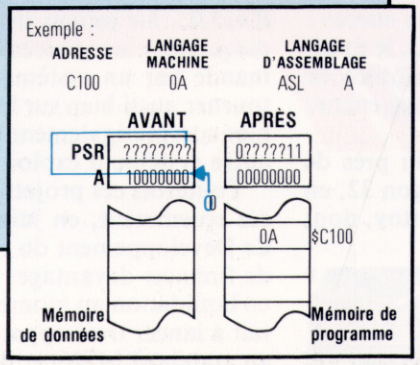
EFFET SUR LE PSR

	S	V		B	D	I	Z	C
MSB	X						X	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	0A	ASL A



SLA — DÉCALER A GAUCHE — Z80

Registre – **CB** (2 octets)

Le contenu de l'octet est décalé d'un bit à gauche en passant par la retenue ; le zéro est décalé dans lsb.

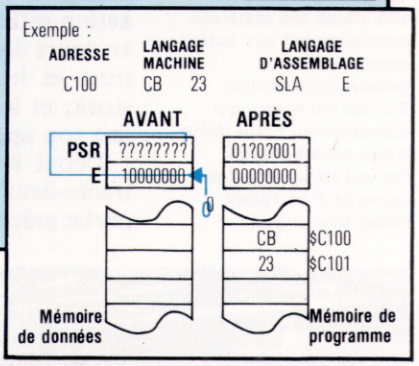
EFFET SUR LE PSR

	S	Z	H		V	N	C
MSB	X	X	0		X	0	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	CB 23	SLA E



CMP — COMPARER L'ACCUMULATEUR — 6502

Absolu – **CD** (3 octets)

Le contenu de l'adresse mémoire est comparé à l'accumulateur.

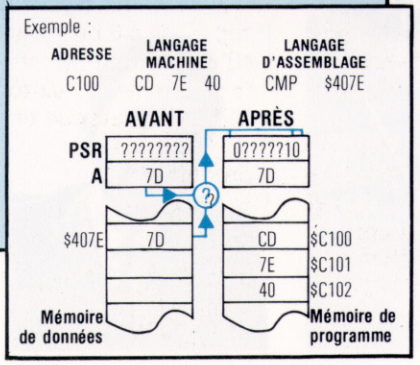
EFFET SUR LE PSR

	S	V		B	D	I	Z	C
MSB	X						X	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	CD 7E 40	CMP $407E



CP — COMPARER L'ACCUMULATEUR — Z80

Immédiat – **FE** (2 octets)

Le contenu de l'octet suivant l'opc est comparé à l'accumulateur.

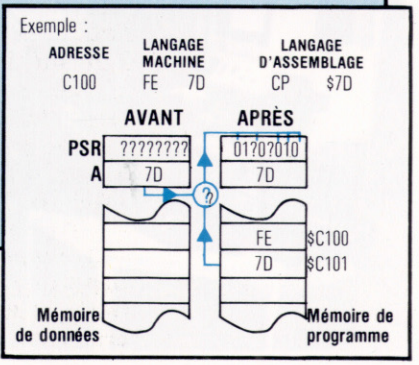
EFFET SUR LE PSR

	S	Z	H		V	N	C
MSB	X	X	X		X	1	X

LSB

Exemple :

ADRESSE	LANGAGE MACHINE	LANGAGE D'ASSEMBLAGE
C100	FE 7D	CP $7D



Tout feu tout flammes

Depuis son lancement en 1982, le Dragon 32 s'est acquis une bonne réputation. Mais de graves problèmes financiers compromettent l'avenir de la compagnie.

Dragon Data fut créée en 1981 par Mettoy, une marque de jouets. Son intention était de tirer profit du boom de la micro-informatique, qui, en Grande-Bretagne, n'en était encore qu'à ses débuts. Grâce à l'aide financière de l'Agence de Développement du Pays de Galles, une usine fut mise en route à Swansea, et le Dragon 32 apparut en août 1982.

Il était construit autour du microprocesseur 6809 de Motorola, et non du Z80 ou du 6502, comme la plupart de ses concurrents. Les circuits imprimés de l'appareil étaient même si conformes aux recommandations de Motorola qu'on accusa Dragon Data d'avoir purement et simplement copié le Color Computer de Tandy, qui lui aussi faisait usage du 6809. Les utilisateurs eurent tôt fait de découvrir que certains programmes pouvaient tourner indifféremment sur les deux machines.

Le Dragon 32 bénéficiait d'un BASIC Microsoft (le dialecte BASIC le plus répandu) et d'un véritable clavier, type machine à écrire. A l'époque de son lancement, le Vic-20 était son seul concurrent dans cette gamme de prix. Un marketing efficace fut aussi à l'origine du succès : au cours des mois précédant Noël 1982, le Spectrum et le BBC Micro étaient en rupture de stock, et le Commodore 64 n'avait pas encore fait son apparition.

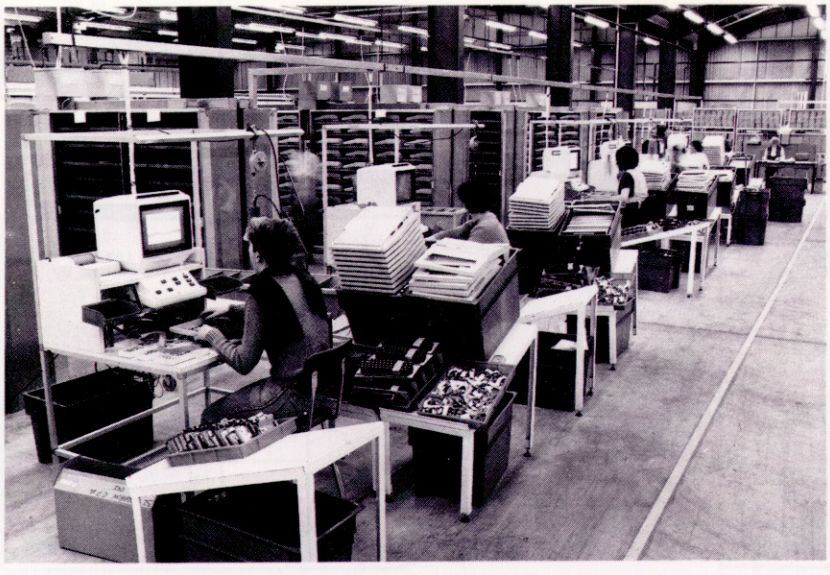
Début 83 la firme avait déjà vendu près de trente-deux mille exemplaires du Dragon 32, en partie grâce au réseau de vente de Mettoy, dont de grands magasins comme Boots et Dixons étaient depuis longtemps de fidèles clients.

Dragon Data dut pourtant faire face à de très graves difficultés financières pendant l'été 83 : Mettoy fut mis en liquidation judiciaire, et l'avenir de sa filiale galloise paraissait compromis. Le salut vint finalement d'un consortium de compagnies menées par Prutec, une firme dépendant de Prudential, une grosse compagnie d'assurances qui s'aventurait ainsi dans le domaine de la technologie avancée. On parvint à réunir l'équivalent de trois millions de francs, à titre de « bouée de sauvetage », et Brian Moore, ancien responsable de GEC, devint directeur de la compagnie. Celle-ci put ainsi surmonter ses problèmes de liquidités, investir dans une nouvelle unité de production à Port Talbot, et poursuivre la mise au point du Dragon 64 et d'un lecteur de disquettes.

Un avenir compromis

Le Dragon 64 dispose de 64 K de RAM, d'un clavier très amélioré et d'une interface série RS232C. Le lecteur de disquettes fait usage de disquettes cinq pouces classiques, et il est commandé par un système d'exploitation qui peut tourner aussi bien sur le 32 que sur le 64. Ce dernier accepte également une version de l'OS9, un autre système d'exploitation très puissant.

Toutefois ces projets furent gravement menacés récemment, en juin 84. Prutec et l'Agence de Développement du Pays de Galles ont refusé de financer davantage. La compagnie était mise en liquidation au moment même où elle s'apprêtait à lancer trois appareils nouveaux, dont l'un au standard MSX, et d'autres produits encore. Un arrangement semble avoir été trouvé : Tandy reprenant le 32 et le 64 et GEC se chargeant des futures machines. Mais le Dragon est désormais une espèce menacée.



La naissance du Dragon
Dragon Data construit elle-même ses appareils, contrairement aux autres constructeurs britanniques comme Sinclair ou Acorn, qui abondonnent cette tâche à des sous-traitants. On voit ici la nouvelle usine de Port Talbot (West Glamorgan).

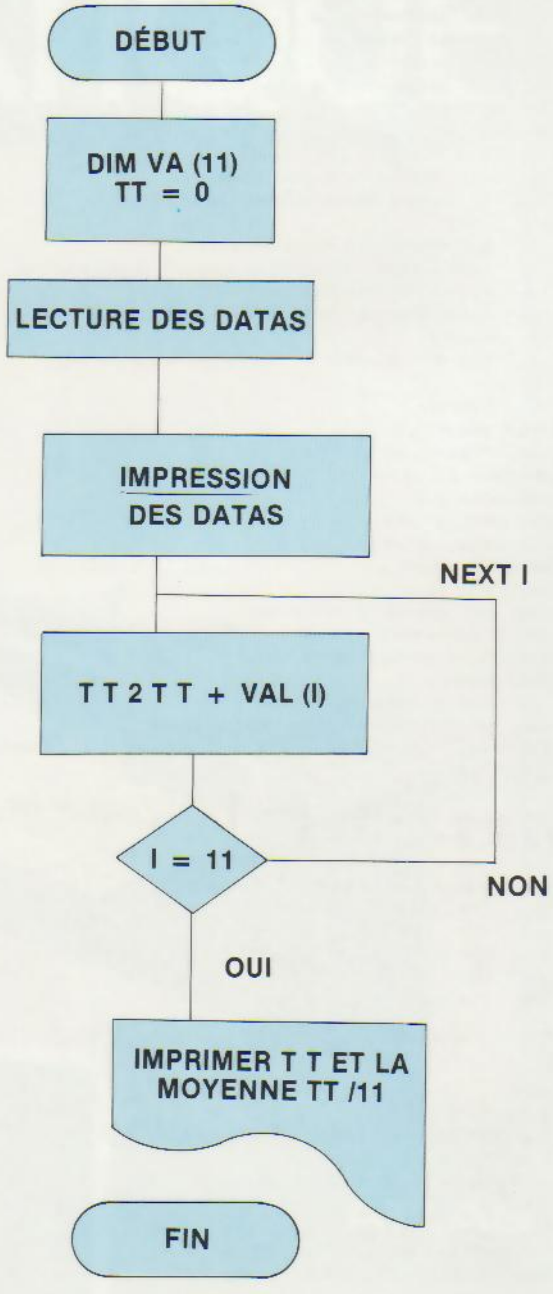
Un homme de talent
Richard Wadman, directeur du marketing de Dragon Data, s'apprêtait à lancer sur le marché une nouvelle gamme d'ordinateurs lorsque la compagnie se retrouva en liquidation.

PROGRAMME N° 24

TRI ET FACTURE

Les traitements effectués dans le programme de tri de la table VA que nous avons vu précédemment peuvent se faire dans deux tables simultanément. Par exemple, pour obtenir la liste des notes dans l'ordre croissant accompagnée de celle des élèves correspondant, il suffit de faire subir à la table EL$ le même traitement que celui de la table VAIII. On obtient alors le programme suivant :

```
ÜLIST

5   REM  ON DIMENSIONNE VA
10   DIM VA(5)
15   REM  ON DIMENSIONNE EL$
20   DIM EL$(5)
25   REM  BOUCLE DE SAISIE DES NOTES ET NOMS DES ELEVES
30   FOR I = 1 TO 5
39   PRINT
40   INPUT "VALEUR ?:";VA(I)
50   PRINT
60   INPUT "NOM DE L'ELEVE ?:";EL$(I)
64   NEXT I
70  TM = 0
80   FOR I = 1 TO 5 - 1
85   REM  BOUCLE DE COMPARAISON
90   IF VA(I + 1) < VA(I) THEN Z = VA(I):VA(I) =
VA(I + 1):VA(I + 1) = Z:Z$ = EL$(I):EL$(I) = EL$(I + 1)
:EL$(I + 1) = Z$:TM = 1
150   NEXT I
160   IF TM <> 0 THEN 70
170   FOR I = 1 TO 5
180   PRINT VA(I),EL$(I)
190   NEXT I


ÜRUN

VALEUR ?:14

NOM DE L'ELEVE ?:DUPONT

VALEUR ?:10

NOM DE L'ELEVE ?:DURAND

VALEUR ?:19

NOM DE L'ELEVE ?:LEMEILLEUR

VALEUR ?:2

NOM DE L'ELEVE ?:LEMAUVAIS

VALEUR ?:14

NOM DE L'ELEVE ?:MARTIN
2              LEMAUVAIS
10             DURAND
14             DUPONT
14             MARTIN
19             LEMEILLEUR
```



Étudions maintenant un petit programme permettant d'éditer une facture. En datas, on va définir des produits (références, désignations, libellés et prix).
Ces produits facturés sont stockés dans quatre tables :

RE$( )	Références
LI$( )	Libellés
PX( )	Prix
Q( )	Quantité

RE$	LI$	PX	Q
PA 1	Papier	0,10	100
CR 1	Crayon	2,50	50
etc.			

Exécution recherchée :

Nom client ? Durand
Adresse ? 10, av. Vétérinaire
Produit (0 pour fin) ? PA 1
Quantité 10
Produit (0 pour fin) ? CR 1
Quantité, 5
etc.

Facture du client : Durand
10, av. Vétérinaire

PA 1	papier	0,10	10	1,00
CR 1	crayon	2,50	5	12,50
	TOTAL			13,50

```
ÜLIST

10   REM  PG DE FACTURATION
20   REM  SAISIE EN DATAS DES REFERENCES,LIBELLES ET PRIX
DES PRODUITS
30   DATA  PA1,PAPIER,0.10
40   DATA  CR1,CRAYON,2.5
50   DATA  ST1,STYLO,3.5
60   DATA  CA1,CAHIER,6.75
65   DATA  0
70   REM  SAISIES ANNEXES
80   INPUT "NOM DU CLIENT :";NM$
90   INPUT "ADRESSE DU CLIENT :";AD$
100   REM  INITIALISATION DU COMPTEUR DE LIGNES
110  CL = 0
120   REM  SAISIES DES PRODUITS VENDUS
130   PRINT : INPUT "REFERENCE DU PRODUIT :(0 POR FIN)";R$
140   IF R$ = "0" THEN 270
150   INPUT "QUANTITE :";Q
170   RESTORE
180   READ RF$: IF RF$ = "0" THEN PRINT "N'EXISTE PAS":
GOTO 130
190   READ LI$
200   READ PX
210   IF R$ < > RF$ THEN 180
220  CL = CL + 1
225  Q(CL) = Q
230  RF$(CL) = RF$
240  LI$(CL) = LI$
250  PX(CL) = PX
260   GOTO 130
270   REM  EDITION DE LA FACTURE
280   PRINT : PRINT : PRINT
290   PRINT "ETABLISSEMENTS DUROC"
300   PRINT
310   HTAB 15: PRINT "FACTURE DE :";NM$
320   PRINT : HTAB 15: PRINT "ADRESSE :";AD$
330   PRINT : PRINT
340  T = 0
350   FOR K = 1 TO CL
360   PRINT RF$(K);
370   HTAB 5: PRINT LI$(K);
380   HTAB 20: PRINT PX(K);
390   HTAB 28: PRINT Q(K);
400   HTAB 35: PRINT Q(K) * PX(K)
405  T = T + Q(K) * PX(K)
410   NEXT K
420   PRINT
430   HTAB 25: PRINT "TOTAL :";T
```

```
ÜRUN
NOM DU CLIENT :DURAND
ADRESSE DU CLIENT :10 AV VETERINAIRE

REFERENCE DU PRODUIT :(0 POR FIN)CA1
QUANTITE :50

REFERENCE DU PRODUIT :(0 POR FIN)CR1
QUANTITE :20

REFERENCE DU PRODUIT :(0 POR FIN)PA1
QUANTITE :100

REFERENCE DU PRODUIT :(0 POR FIN)ST1
QUANTITE :120

REFERENCE DU PRODUIT :(0 POR FIN)0


ETABLISSEMENTS DUROC

              FACTURE DE :DURAND

              ADRESSE :10 AV VETERINAIRE

CA1 CAHIER          6.75     50     337.5
CR1 CRAYON          2.5      20     50
PA1 PAPIER          .1       100    10
ST1 STYLO           3.5      120    420

                        TOTAL :817.5
```